Recibido 02 Nov 2025 *ReCIBE, Año 15 No. 1, mayo 2025*

Aceptado 22 Abr 2026

Prácticas de Ciberseguridad en la Ingeniería de Software: Una Revisión de la Literatura

Cybersecurity Practices in Software Engineering: A Literature Review

Eduardo Antonio Castillo Garrido 1
e.antonio.cg@outlook.com
Juan Carlos Pérez-Arriaga 1
juaperez@uv.mx
Héctor Xavier Limón-Riaño 1
hlimon @uv.mx
Saúl Domínguez-Isidro 1
sauldominguez @uv.mx

[1] *Facultad de Estadística e Informática,*
Universidad Veracruzana, Xalapa, Veracruz, México.

Resumen

El desarrollo de productos de software que consideren aspectos de ciberseguridad en el diseño sigue representando un reto considerablemente alto para los equipos encargados de su construcción. En la Ingeniería de Software se han implementado diversas prácticas en distintas fases del desarrollo de software que tienen como objetivo construir productos cada vez más seguros, con la finalidad de satisfacer las demandas de la industria. A través de una revisión sistemática de 77 artículos, hemos identificado 30 prácticas relevantes de ciberseguridad, incluyendo políticas de seguridad, modelado de amenazas, análisis estático y pruebas de penetración. También se destacan artefactos clave como diagramas de clases, casos de mal uso y patrones de seguridad, junto con herramientas, modelos, estándares y marcos asociados. Estos hallazgos muestran cómo la ciberseguridad se integra en todas las etapas del ciclo de vida del desarrollo de software, con la finalidad de producir productos de software que satisfagan uno de los atributos de calidad más relevantes en la actualidad.

Palabras Clave: Software Seguro; SDLC; Desarrollo de Software Seguro; Ingeniería de Software; Revisión Sistemática de la Literatura; Síntesis Narrativa.

Abstract

Developing software products that incorporate Cybersecurity into their design remains a relatively hard challenge for development teams. Software Engineering has implemented several practices at different stages of the software development lifecycle, aimed at building increasingly secure products to meet industry demands. Through a systematic review of 77 articles, we identified 30 relevant cybersecurity practices, including security policies, threat modeling, static analysis, and penetration testing. Key artifacts such as class diagrams, misuse cases, and security patterns are also highlighted, along with associated tools, models, standards, and frameworks. These findings demonstrate how cybersecurity is integrated into all stages of the software development lifecycle to produce software products that meet one of today's most important quality attributes.

Keywords: Secure Software; SDLC; Software Secure Development; Software Engineering; Literature Review Synthesis, Narrative Synthesis.

Introducción

Hoy en día el software representa una parte esencial para la industria, la economía, el entretenimiento, las comodidades cotidianas y muchas otras áreas que antiguamente estaban desvinculadas de la tecnología. Sin embargo, el crecimiento constante de estas tecnologías trae consigo problemas de ciberseguridad. Casos como el de ION Group en 2023 (Lemos, 2023) son un ejemplo del impacto que las amenazas de ciberseguridad tienen en los mercados globales y que consecuentemente pueden afectar a gran parte de la población en el mundo. Este dato cobra más relevancia al saber que al menos para el año 2022, el 60% de las corporaciones del mundo almacenaban sus datos en la nube (Vailshery, 2022). Estos hechos refuerzan la necesidad de proteger el desarrollo de sistemas seguros ante las actuales y futuras amenazas cibernéticas (Vijayan, 2023).

Dado que el software constituye un área dominante dentro de la dependencia tecnológica actual, la ciberseguridad en el desarrollo de software puede contribuir a mitigar futuros ataques a los sistemas y prevenir la pérdida de activos importantes.

Para los ingenieros de software, la seguridad es una tarea crucial, ya que implica satisfacer una necesidad que no siempre se reconoce de manera consciente. Generalmente un ingeniero de software emplea metodologías que ayudan a crear productos de software de manera sistemática, de modo que el resultado cumpla con las expectativas de funcionalidad y calidad establecidas por los

interesados (stakeholders) (Sommerville, 2011). No obstante, es bien sabido que las metodologías de desarrollo clásicas tienden a dejar el aspecto de la seguridad hasta el final de las actividades críticas (Emami et al., 2010), lo que representa un riesgo para quienes desconocen su importancia real.

Straub (2020) destaca que la ingeniería de software constituye la primera línea de defensa contra los ciberataques y encontró que diversos autores proponen técnicas enfocadas en gestionar la seguridad durante el proceso de desarrollo de software. Estos autores también abogan por la enseñanza de la ciberseguridad como parte de los programas de ingeniería de software. El autor enfatiza que es necesario identificar los requisitos de seguridad y gestionarlos a lo largo del desarrollo, el despliegue y el mantenimiento del software. Por esta razón, han surgido diversas metodologías que buscan implementar la seguridad en el desarrollo de software, y cuyos enfoques se han ido incorporando gradualmente en programas educativos relacionados con las ciencias de la computación (González et al., 2019).

Shift Left Security, por ejemplo, es un enfoque con el cual se pretende incorporar la seguridad a lo largo de todo el ciclo de vida del desarrollo de software. El objetivo es trasladar la seguridad al inicio del proceso de desarrollo para priorizar las prácticas de seguridad antes de que tengan lugar las actividades típicas de desarrollo. Mientras tanto, un enfoque de DevSecOps promueve la comunicación y colaboración entre equipos de desarrollo, operaciones y equipos de seguridad. Al mismo tiempo que involucra un cambio de cultura de trabajo, al priorizar la seguridad a lo largo del proceso de desarrollo de software (Abiona et al., 2024). Ejemplos del desarrollo de software seguro incluyen modelos como Secure Software Development Life Cycle (SSDLC), el cual es un enfoque que integra la seguridad en los procesos del desarrollo de software y que incluye prácticas de seguridad como el desarrollo de requisitos de seguridad, el análisis de riesgos y las pruebas de seguridad transversalmente a las etapas correspondientes en el SDLC (Check Point, 2024).

Por lo tanto, la ingeniería de software representa uno de los medios más adecuados para implementar prácticas de ciberseguridad, de modo que los ingenieros de software puedan crear aplicaciones y otros productos que satisfagan las necesidades de seguridad actuales (González et al., 2019).

El resto del documento se organiza de la siguiente manera: la Sección 2 aborda los trabajos relacionados como antecedentes. La Sección 3 explica el método de investigación utilizado para llevar a cabo el estudio. La Sección 4 describe el proceso de la investigación. La Sección 5 presenta los resultados obtenidos. La Sección 6 discute los hallazgos y sus implicaciones. La Sección 7 expone las amenazas a la validez de nuestro estudio. Finalmente, la Sección 8 concluye el artículo con un resumen de las contribuciones y sugerencias para investigaciones futuras.

Trabajos relacionados

En los últimos años, ha crecido el interés por incluir prácticas de seguridad en el desarrollo de software. Algunos estudios previos abordan enfoques de desarrollo similares al nuestro y, a su vez, demuestran que aún existen áreas de oportunidad para la exploración de temas relacionados con la ciberseguridad y el desarrollo de software.

En el primero de los estudios relacionados, Khan et al. (2020) investigaron el estado del arte de las prácticas de seguridad en software, identificando hasta 37 procesos, modelos y frameworks relacionados con la implementación de seguridad en la ingeniería de software. Aunque este artículo no especifica prácticas de ciberseguridad en particular, resulta un punto de partida útil para considerar estándares y modelos a seguir con el fin de incluir prácticas ampliamente aceptadas en

la industria. Nuestro trabajo, por otra parte, identifica prácticas de ciberseguridad específicas del desarrollo de software y reporta frameworks y modelos relacionados con estas prácticas.

En el trabajo de González et al. (2021) se aborda la necesidad de concientizar a los estudiantes de ingeniería de software sobre la inclusión de la seguridad en las etapas de la ingeniería de requisitos. Propusieron una guía con 14 requisitos y sugerencias para incluir en cursos de desarrollo de software, enfocándose en buenas prácticas de Ciberseguridad y consideraciones para mantener un entorno de trabajo seguro. También sugieren considerar modelos como SSDLC y OWASP Top 10; sin embargo, los autores aclaran que no proponen prácticas específicas alineadas con el ciclo de vida del desarrollo de software. Aunque en este trabajo se trata la concienciación de la Ciberseguridad en los estudiantes de nivel universitario, la mayoría de las consideraciones propuestas por los autores se relacionan con buenos hábitos en el entorno de desarrollo con prácticas de sanitización e higiene. Nuestro trabajo se centra en explorar aquellas prácticas relacionadas con los procesos de la ingeniería del software mediante la clasificación de las prácticas entre las etapas del desarrollo de software.

Posteriormente Khan et al. (2022a) identifican prácticas de seguridad asociadas a un conjunto de riesgos de seguridad. Los autores han reportado 423 prácticas para el desarrollo de software seguro incluyendo las etapas de despliegue y mantenimiento de software. Adicionalmente Khan et al. (2022b) reportan un conjunto de métricas de seguridad relacionadas con la Ingeniería de Software segura. También han reportado herramientas, estándares, y algunos temas de investigación asociados con las prácticas para la seguridad del software, sin embargo, no reportan prácticas específicas. Nuestro trabajo, además de reportar prácticas, también abarca los hallazgos sobre artefactos, herramientas, estándares, modelos de referencia, frameworks e incluso recursos de información de ciberseguridad asociados a las mismas. Además, hemos realizado la asociación entre tales prácticas y herramientas, por lo que es más sencillo diferenciar en qué etapas es adecuado considerar cada una de estas.

Finalmente, Selva-Mora & Quesada-López (2024) realizaron un estudio sobre prácticas de seguridad en el desarrollo ágil. En su investigación, reportan al menos 12 prácticas clave, categorizadas en las etapas de desarrollo del modelo BSIMM. Al mismo tiempo, reportan diversos beneficios de estas prácticas y los diferentes desafíos relacionados con la implementación de estas. Nuestro trabajo, por su parte, reporta un mayor número de prácticas y elementos relacionados con las mismas, los cuales mencionamos antes. Además, en el apartado de discusión de este trabajo hemos analizado los desafíos relacionados con la integración de prácticas de ciberseguridad en el desarrollo de software actual.

A partir de la identificación y clasificación de los hallazgos de este trabajo, se consolida una base documental de referencia que permita a interesados en el tema, profundizar en el diseño de estrategias enfocadas en incluir una perspectiva de ciberseguridad en procesos de desarrollo de software y enseñanza de la ingeniería de software.

Método de investigación

Esta investigación se ha desarrollado mediante una Revisión Sistemática de la Literatura (RSL) siguiendo el método propuesto por Kitchenham et al. (2015).

La metodología de Kitchenham es ampliamente reconocida como el estándar de facto para conducir revisiones sistemáticas en el ámbito de la ingeniería de software. Su adopción se justifica por ofrecer un proceso estructurado y reproducible, compuesto por tres fases claramente definidas:

planificación, ejecución y documentación de la revisión. Este enfoque garantiza la trazabilidad, transparencia de los resultados y aspectos esenciales para mantener la validez científica del estudio.

Si bien existen otras propuestas metodológicas para revisiones sistemáticas, el método de Kitchenham se distingue por su adaptación específica a la investigación empírica en ingeniería de software, a diferencia de enfoques más generales provenientes de otras disciplinas. Además, su amplio uso en la comunidad científica permite comparar y validar resultados con estudios previos, fortaleciendo la consistencia metodológica de esta investigación.

Preguntas de investigación (PI)

Se definieron tres preguntas de investigación que proporcionan una comprensión del estado del arte sobre las principales prácticas de Ciberseguridad utilizadas en el desarrollo de software, incluyendo la identificación de artefactos y diversos aspectos orientados a implementar dichas prácticas.

Pregunta de Investigación	Objetivo
PI-1: ¿Qué prácticas de Ciberseguridad se utilizan en la ingeniería de software?	Identificar las prácticas, actividades o técnicas de Ciberseguridad aplicadas en procesos de ingeniería de software.
PI-2: ¿Cuáles son las fases en las que se aplican las prácticas de Ciberseguridad?	Determinar las fases del SDLC en las que se integran dichas prácticas.
PI-3: ¿Qué tipo de artefactos se producen al realizar prácticas de Ciberseguridad?	Reconocer los artefactos de software comúnmente generados por estas prácticas.

Tabla 1. Preguntas de Investigación

Proceso de Búsqueda

El proceso de búsqueda para identificar estudios relevantes en este campo estuvo guiado por el método de Zhang et al. (2011). Realizamos una búsqueda manual limitada a un pequeño conjunto de estudios relevantes de diversas conferencias relacionadas con Ciberseguridad y desarrollo de software. Con base en estos estudios, se seleccionó un conjunto de palabras clave con las cuales se desarrolló la cadena de búsqueda, misma que a su vez fue evaluada mediante las métricas de **Recall** y **Precision** del método *Quasi-Gold Standard* (Zhang et al., 2011) utilizando los estudios de la búsqueda manual. La cadena de búsqueda fue modificada hasta alcanzar una versión final con un valor recall de 0.8 y una precisión de 0.0073 en la búsqueda de dichos artículos.

	AND		
OR	Software Engineering	Cyber Security	Practice
	Development Process	Computer Security	Approach
	SDLC	Secure Software	Technique
	Software Development	Software Security	Method
	Software Development Life Cycle	DevSecOps	Paradigm
			Framework

Tabla 2. Términos de Búsqueda

Cadena Final:

"Software Engineering" **OR** "Development Process" **OR** "SDLC" **OR**
"Software Development" **OR** "Software Development Life Cycle"
AND
"Cyber Security" **OR** "Computer Security" **OR** "Secure Software" **OR**
"Software Security" **OR** "DevSecOps"
AND
"Practice" **OR** "Approach" **OR** "Framework" **OR** "Method" **OR** "Paradigm" **OR**
"Technique"

Proceso de Selección

Se definieron los criterios de inclusión y exclusión (ver Tabla 3) con el fin de realizar una selección efectiva de artículos. Dichos criterios se dividieron en cuatro fases de la revisión sistemática de la literatura, presentadas en la Tabla 6 de la siguiente sección.

Criterios de Inclusión	Criterios de Exclusión
CI-1: El artículo está en inglés	CE-1: No hay acceso al documento
CI-2: La fecha de publicación está entre 2018 y 2024	CE-2: El documento no es un artículo de investigación
CI-3: El título o resumen responde a una pregunta de investigación	CE-3: El artículo es un estudio secundario
CI-4: El artículo responde al menos a una pregunta de investigación	CE-4: El artículo está repetido

Tabla 3. Criterios de Elegibilidad

Evaluación de la Calidad

Para llevar a cabo la investigación y la selección de artículos de calidad, se adoptaron algunas de las preguntas propuestas por el enfoque de Kitchenham (2015, pp. 83--85). La puntuación de cada estudio se determinó en función del cumplimiento de cada pregunta: 1 punto si se cumplía, 0.5 puntos si se cumplía parcialmente, y 0 puntos si no se cumplía. Las preguntas para evaluación de la calidad se enlistan a continuación:

1	¿Está el artículo basado en una investigación (o es meramente un reporte basado en 'lecciones aprendidas' bajo la opinión de un experto)?
2	¿Se enuncian claramente los objetivos de la investigación?
3	¿Se describe adecuadamente el contexto en el que se llevó a cabo la investigación?
4	¿El diseño de la investigación fue adecuado para abordar los objetivos de la investigación?
5	¿El análisis de los datos fue lo suficientemente riguroso para los objetivos de la investigación?
6	¿Se reportan claramente los hallazgos del estudio?
7	¿Se comunica adecuadamente a la audiencia las conclusiones, las implicaciones para la práctica y la investigación futura?

Tabla 4. Preguntas de Calidad

Proceso de Extracción y Síntesis de Datos

El formato de extracción de datos, disponible en el Anexo C, permitió separar los hallazgos de cada artículo en relación con las preguntas de investigación. Para analizar los hallazgos, sintetizamos la información mediante un enfoque de Síntesis Narrativa (*narrative synthesis*), de acuerdo con Popay et al. (2006). Este enfoque incluye el desarrollo de una teoría de cambio, la tabulación de los datos, el análisis de relaciones entre ellos y la evaluación de la solidez de la síntesis.

DATOS GENERALES	
Título	
Autor(es)	
Año	
Fuente	
Tipo de Publicación	

Referencia	
Palabras clave	
Abstract o resumen	
Pregunta de investigación que responde	PI-1 Práctica:
	PI-2 Etapa del desarrollo:
	PI-3 Artefactos:
Notas	
ANÁLISIS GENERAL DEL ESTUDIO	

Tabla 5. Formato de extracción de información de los estudios primarios

Conducción de la revisión

El proceso de búsqueda consistió en aplicar los criterios de elegibilidad a todos los registros recuperados de las bibliotecas digitales seleccionadas: IEEE Xplore, ACM Digital Library, SpringerLink y ScienceDirect, a través de las siguientes etapas:

- Etapa 0: Se ejecutó la cadena de búsqueda sin criterios de elegibilidad.
- Etapa 1: Se excluyeron aquellos registros que no cumplían con los criterios básicos de inclusión: publicaciones en inglés, con acceso al texto completo y publicadas entre 2018 y 2024.
- Etapa 2: Se excluyeron los documentos que no eran considerados artículos de investigación primaria.
- Etapa 3: Se excluyeron los estudios secundarios como revisiones sistemáticas y las entradas duplicadas. Se incluyeron los estudios cuyos títulos y resúmenes mostraban indicios de responder al menos a una de las preguntas de investigación.
- Etapa 4: Se incluyeron aquellos artículos cuyo contenido respondía de manera efectiva a una o más de las preguntas de investigación definidas. La cantidad de artículos durante el proceso de filtrado se resume en la Tabla 6.

Etapas	IEEE Xplore	ACM DL	Springer	Science Direct	Total
0	1954	11949	7789	3638	25330
1	1160	6405	3367	2189	13121
2	1107	400	1924	1762	5193
3	96	28	33	44	201
4	40	14	10	13	77

Tabla 6. Resultado por cada etapa de búsqueda

Para la evaluación de calidad disponible en el Anexo B, la puntuación promedio de los estudios alcanzó los 6.53 de un total de 7 puntos. Además, 18 estudios obtuvieron la puntuación máxima. La deficiencia más común en los estudios se debió a la falta del contexto especifico en el que se llevaron a cabo los estudios. Este resultado indica un alto nivel de calidad metodológica en los estudios seleccionados.

Resultados

Se revisaron 77 estudios publicados entre 2018 y 2024, disponibles en el Anexo A,. Los años con mayor número de estudios fueron 2019, 2023 y 2024 (Ver Figura 1). El 52% de estos artículos provienen de congresos, mientras que el 48% restante corresponde a artículos de journals. IEEE Xplore es la fuente con la mayor distribución de artículos, representando 40 de los 77 artículos en total. De las demás fuentes, se incluyen 14 estudios de la ACM Digital Library, 13 de Science Direct y 10 de Springer Link (Ver Figura 2). Si bien IEEE Xplore presenta la mayor concentración de estudios, comparte publicaciones de conferencias con la ACM Digital Library, por lo que algunos estudios duplicados fueron excluidos de ACM.

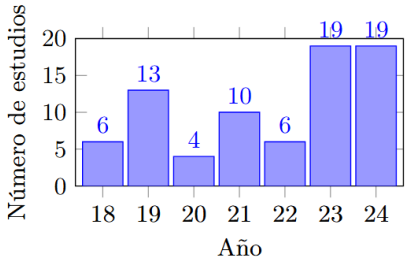

Figura 1. Distribución de los estudios hallados por año de publicación

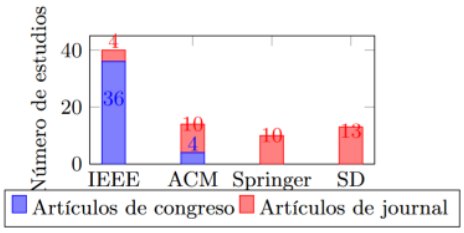

Figura 2. Tipos de publicaciones por fuente

Respuesta a las Preguntas de Investigación (PI) 1 y 2

Para responder a las Preguntas de Investigación 1 y 2 (PI-1: ¿Qué prácticas de Ciberseguridad se utilizan en la ingeniería de software?; PI-2: ¿En qué fases del ciclo de vida del desarrollo de software se aplican?), los hallazgos reportados en la Tabla 7 muestran que las prácticas identificadas se aplican a lo largo de todo el ciclo de vida, con mayor concentración en las etapas iniciales. La aplicación de políticas de seguridad asegura el cumplimiento de estándares y establece controles organizacionales, mientras que la gestión de riesgos, el modelado de casos mal uso / uso

indebido (Misuse Cases) y las historias de usuario permiten recopilar y definir requisitos de seguridad. En la fase de diseño, la práctica Threat Modeling evalúa y aborda riesgos. Durante el desarrollo, el análisis estático de código, la revisión de código y las pruebas unitarias e integrales controlan la calidad y detectan vulnerabilidades tempranas. Finalmente, las pruebas de software, incluyendo carga, estrés, rendimiento y estructurales, buscan vulnerabilidades que solo se manifiestan en ejecución, asegurando la resiliencia del software en entornos exigentes.

Práctica	ID del Artículo	Frec.	Fase SDLC
Threat Modeling	1, 4, 6, 7, 8, 10, 11, 15, 19, 20, 25, 33, 36, 37, 38, 42, 43, 47, 48, 59, 60, 66, 67, 68, 69, 70	26	Software Design
Risk Management	1, 2, 3, 4, 5, 7, 17, 20, 22, 31, 33, 35, 43, 47, 49, 50, 55, 65, 67, 71	19	Software Requirements
Static Code Analysis	7, 11, 13, 18, 23, 26, 27, 30, 34, 36, 37, 39, 40, 41, 65, 67, 69, 70	18	Software Construction
Apply Security Policies and Regulations	2, 4, 5, 7, 13, 17, 20, 21, 28, 32, 33, 43, 55, 56, 65	15	All Phases
Model Misuse Cases	6, 9, 12, 16, 19, 20, 23, 31, 34, 35, 39, 67	14	Software Design
Code Review	4, 11, 19, 20, 21, 28, 30, 31, 41, 49, 65, 67	12	Software Construction
Design Security Architecture	4, 5, 9, 1, 25, 27, 56, 66, 67, 70	10	Software Design
Penetration Testing	4, 6, 13, 23, 27, 28, 33, 36, 45	9	Software Testing
Continuous Monitoring	3, 14, 20, 21, 25, 26, 33, 47, 56	9	Software Construction & Testing
Dynamic Code Analysis	13, 26, 34, 36, 38, 39, 40, 69	8	Software Construction
Model Abuse Cases	2, 6, 12, 20, 23, 67, 70	7	Software Requirements
Automating Testing	11, 15, 16, 21, 32, 67, 36	7	Software Construction & Testing
Design Review	4, 5, 11, 15, 20, 21	6	Software Design
Security Audits	4, 6, 11, 20, 41, 66	6	All Phases

Include Coding Standard / Coding Rules	3, 6, 15, 17, 67	5	Software Construction
Fuzz Testing	20, 21, 28, 65	4	Software Testing
Apply metrics to software evaluation	3, 21, 27	3	All Phases
Separation of Roles	4, 20, 24	3	Software Construction & Testing
Unit Testing	4, 21	2	Software Construction
Use of Version Control System	16, 49	2	Software Construction
Include Security Patterns	44, 56	2	Software Design
Integration Testing	16, 21	2	Software Testing
Privacy Testing	4, 21	2	Software Testing
Simulate Attack	4, 39	2	Software Testing
Validate Input and Outputs	14	1	Software Construction & Testing
Regression Testing	21	1	Software Testing
Load Tests	21	1	Software Testing
Stress Test	21	1	Software Testing
Performance Test	21	1	Software Testing
Structural Testing	21	1	Software Testing

Tabla 7. Prácticas de Ciberseguridad

Respuesta a la Pregunta de Investigación (PI) 3

Para responder a la PI-3 (¿Qué tipo de artefactos se producen al aplicar prácticas de Ciberseguridad?), los resultados mostrados en la Tabla 8 indican que el artefacto más recurrente es el Misuse Case Diagram, vinculado con los diagramas de casos de uso y presente en el 22.5% de los estudios. Otros artefactos destacados incluyen, en la etapa de requisitos, el Abuse Case Diagram, el Context Diagram y el Domain Model; y en la etapa de diseño de software, diagramas de clases, diagramas de flujo de datos y de secuencia, árboles de ataque y varios diagramas de patrones de diseño. En las etapas posteriores del desarrollo no se identificaron artefactos

específicos, salvo el uso implícito de diagramas de flujo de datos en pruebas durante la construcción.

Artefacto	ID del Artículo	Frec.	Fase SDLC
Misuse Cases / Abuse Case Diagram	2, 6, 9, 12, 16, 19, 20, 23, 31, 34, 35, 39, 41, 67, 74	15	Software Requirements & Software Design
Use Case Diagram	2, 9, 12, 16, 31, 34, 39, 44, 56	9	Software Design
Class Diagram	2, 9, 20, 44, 56, 69, 70	7	Software Design
Data Flow Diagram	1, 8, 10, 42, 70	5	Software Design
Sequence Diagram	2, 56, 44, 69	4	Software Design
Checklists	4, 13, 69	3	Verification & Validation (V&V)
Context Diagram	1, 2	2	Software Requirements
Activity Diagram	44, 56	2	Software Design
Abstract Security Pattern	44, 56	2	Software Design
Architectural Security Pattern Diagram	44, 56	2	Software Design
Security Solution Frames	44, 56	2	Software Design
Threats and Type of Threats List	1	1	Software Requirements
Abuse Histories	19	1	Software Requirements
Malicious User Stories	19	1	Software Requirements
Misuse Patterns	44	1	Software Design
Security Cluster	44	1	Software Design
Cloud Security and Privacy Metamodel	44	1	Software Design
Attack Tree Diagram	55	1	Software Design
State Diagram	56	1	Software Design

Tabla 8. Artefactos producto de la aplicación de Prácticas de Ciberseguridad en los procesos de Ingeniería de Software

Otros Hallazgos

Además de las prácticas y artefactos, este estudio identificó herramientas, estándares, modelos, marcos de trabajo y algunos recursos de información relevantes para la ciberseguridad en el desarrollo de software. La Tabla 9 resume estos hallazgos, destacando normas como ISO 31000, ISO/IEC 27005 y la metodología OCTAVE, que proporcionan lineamientos para la gestión de riesgos. Muchos de estos elementos se aplican a múltiples prácticas y consideran aspectos organizacionales relacionados con la implementación de la Ciberseguridad en la creación de software.

Estándar / Modelo / Framework	ID del Artículo	Frec.
DevSecOps	8, 11, 13, 24, 25, 26, 27, 32, 33, 34, 35, 37, 39, 41, 57, 58, 59, 61, 62	19
Microsoft Security Development Lifecycle (Microsoft SDL)	3, 6, 11, 20, 35, 36, 38, 40, 43, 50	10
Microsoft STRIDE	1, 8, 20, 43, 44, 66, 68, 70	8
BSIMM	5, 6, 7, 11, 20, 35, 50, 66	8
OWASP SAMM	5, 7, 11, 20, 21, 50, 66	7
SAFECode	6, 7, 15, 20, 21, 67	6
OWASP Top Ten	8, 11, 18, 21, 27, 55	6
OWASP CLASP	3, 6, 11, 16, 20, 50	6
Common Weakness Enumeration (CWE)	18, 65, 69, 70	4
SQuaRE (ISO 25000:2005)	5, 21, 35, 50	4
CMMI	3, 4, 5, 35	4
NIST SP 800	3, 17, 22	3
Common Criteria	7, 20, 67	3
ISO/IEC 27001	8, 21, 43	3
ISO/IEC 25010	16, 21, 43	3
SSDLC / Secure SDLC	4, 36	2
Secure TROPOS	5, 35	2

SDL Agile	6, 11	2
Common Vulnerabilities and Exposures (CVE)	18, 69	2
Software Security Touchpoints	20, 67	2
OCTAVE	29, 43	2
NICE Framework	55, 65	2
Microsoft DREAD Model	29, 43	2
ISO 3100	1	1
Security Technical Implementation Guide (STIG)	4	1
CIS Benchmark	4	1
IEC 63443	4	1
Security Requirements Engineering Process (SREP)	5	1
Software Security Approach (SWSec)	6	1
COCOMO II	6	1
IFPUG	6	1
COSMIC Function Points	6	1
SEER-SEM	6	1
Quality Assurance (QA)	8	1
ISMS (Information Security Management System)	8	1
Oracle Software Security Assurance (OSSA)	11	1
TSP-Secure	11	1
OWASP Proactive Controls	14	1
Fisa-XP	16	1
SecSDM	17	1
IEEE 1220-2005	17	1

IATF (Information Assurance Technical Framework)	17	1
ISO/IEC TR 13335-3 (Withdrawn)	17	1
SEED Project (Hands-on Labs)	18	1
OWASP ASVS	19	1
Securosis	20	1
ISO/IEC 15939:2002	21	1
MITRE ATT&CK®	21	1
ISO/IEC/IEEE 15288	22	1
ISO/IEC/IEEE 42010	22	1
ISO/IEC/IEEE 29119	23	1
NICE NCWF	24	1
CAE CD core Kus	26	1
SSE-CMM	28	1
VAHTI	28	1
KATAKRI	28	1
CAPEC	31	1

Tabla 9. Otros Hallazgos

La Tabla 10 presenta la lista de diversas herramientas mencionadas en los estudios, las cuales están asociadas a varias de las prácticas identificadas. Aunque la mayoría de las herramientas se relacionan con prácticas como el análisis estático de código y distintas pruebas de software, algunas resultan útiles para la identificación de riesgos y amenazas de seguridad, como *Protection Poker* y *Elevation of Privilege Card Game*. En cuanto al software seguro, encontramos herramientas como *UMLsec* y *UMLpac*, que permiten a los ingenieros de software crear diseños UML con características de seguridad.

Herramienta	ID del Artículo	Práctica Asociada
UMLsec	2, 6, 9, 23, 35, 71	Security Requirements and Design
Protection Poker	19, 23, 66	Risk Assessment
SonarQube	23, 27, 36	Static Code Analysis
Microsoft Threat Modelling Tool	8, 43	Threat Modeling
FlawFinder	18, 65	Static Code Analysis
CheckMarx	23, 27	Static Code Analysis
OWASP Zed Attack Proxy (ZAP)	23, 36	Penetration Testing
Burp Suite	23, 36	Penetration Testing
UMLpac	2	Security Requirements and Design
Elevation of Privilege (EoP) Card Game	8	Threat Modeling
Vulnerable Software Packages	10	Software Security Training
Loggly by Solar Winds	18	Log Analysis
LowViewPlus	18	Log Analysis
Clang-Tidy	18	Static Code Analysis
HP Fortify SCA	23	Static Code Analysis
Jenkins	23	Static Code Analysis
Threadfix	23	Static Code Analysis
JMeter	23	Load and Performance Testing
Postman	23	Load and Performance Testing
BlazeMeter	23	Load and Performance Testing
Goldeneye	23	Penetration Testing
HTTPerf	23	Performance Testing
SoapUI	23	Web Testing
CA APM	23	Performance Testing
Arachni	23	Penetration Testing

XSS ME	23	Penetration Testing
SQL Injection ME	23	Penetration Testing
Meta Exploit	23	Penetration Testing
NMAP	23	Penetration Testing
Whatweb	23	Penetration Testing
Snyk	24	Static Code Analysis
Retire JS	24	Static Code Analysis
Hakiri	24	Static Code Analysis
PVS-Studio	24	Static Code Analysis
Veracode	24	Static Code Analysis
Coverity	24	Static Code Analysis
Rational Team Concert	33	Security Audit
Rational Quality Manager	33	Software Testing
Bugzilla	69	Error Tracing

Tabla 10. Herramientas para el desarrollo de software seguro

Discusión

Se identificaron un total de 30 prácticas de Ciberseguridad asociadas a las diferentes etapas del Ciclo de Vida del Desarrollo de Software (SDLC, por sus siglas en inglés). Aunque la Tabla 7 muestra una concentración de prácticas vinculadas con las fases de prueba y construcción, las prácticas consideradas más críticas —la gestión de riesgos y el modelado de amenazas— se ubican en las etapas iniciales de requisitos y diseño. Esto coincide con la literatura previa, que subraya la necesidad de incorporar la Ciberseguridad desde las fases tempranas del desarrollo, donde las decisiones arquitectónicas tienen un impacto duradero.

Los *Misuse Cases* (y su variante llamada *Abuse Cases*) constituyen otra práctica altamente citada en los estudios, especialmente en la fase de diseño, donde se utilizan para refinar los requisitos de seguridad. Su estrecha relación con los diagramas de casos de uso tradicionales facilita la identificación de amenazas y comportamientos no deseados, lo que explica su amplia adopción en la literatura revisada. Dado que estas prácticas generan artefactos tangibles, los *Misuse Cases* también fueron catalogados en la tabla de artefactos cada vez que los autores los mencionaron como tales.

Con respecto a los estándares y metodologías, OWASP fue el marco más citado, especialmente mediante sus proyectos específicos (*Top 10, CLASP y SAMM*), lo que evidencia una fuerte orientación práctica en los estudios revisados. También se identificaron referencias a *BSIMM,*

CMMI, NIST e *ISO/IEC 27001*, lo que muestra un esfuerzo por adaptar buenas prácticas corporativas y marcos de gestión de seguridad al ámbito del desarrollo de software. Este hallazgo sugiere una convergencia entre enfoques organizacionales y de ingeniería.

El análisis de herramientas refleja una tendencia creciente hacia la automatización de la seguridad. Herramientas como *UMLsec*, *Protection Poker* y las de análisis estático de código (*FlawFinder, SonarQube, Checkmarx*) muestran que la comunidad busca integrar la seguridad dentro de flujos ágiles y DevSecOps. No obstante, pocos estudios discuten la efectividad de estas herramientas en escenarios reales o su adopción en pequeñas y medianas empresas, lo que representa un vacío en la literatura.

Asimismo, varios trabajos destacaron el uso de métricas para monitorear la calidad y seguridad del software, aunque su aplicación sistemática aún es limitada. Esto apunta a la necesidad de desarrollar indicadores más específicos que permitan evaluar la incorporación de prácticas de Ciberseguridad a lo largo del SDLC.

Finalmente, un hallazgo transversal es la dificultad de integrar efectivamente la Ciberseguridad en los procesos de desarrollo de software actuales. La falta de concienciación y formación en Ciberseguridad, tanto en el área académica como en la industrial, sigue siendo una barrera persistente en los diversos contextos en los que opera el desarrollo de software actual, siendo el contexto organizacional algo que pocos estudios consideran al momento de considerar mejores prácticas.

A pesar de los avances conceptuales y metodológicos, aún existe una brecha entre la teoría y la práctica en la implementación de software seguro. Si bien muchos de los procesos metodológicos recomendados en la literatura contienen pautas generales para la implementación de mejores prácticas, no siempre han sido claras o especificas las condiciones en las que estas fueron efectivas.

Por otro lado, aunque en la literatura existe diversa información sobre mejores prácticas independientemente del rubro dentro del desarrollo de software, es poco frecuente que se mencionen específicamente aquellas malas o peores prácticas que tanto los ingenieros de software como las organizaciones, podrían estar permitiendo sin ser conscientes de ello. Aunque muchas de las mejores prácticas, manejan de manera implícita aquellos malos hábitos en el desarrollo de software, algunos otros podrían no ser tan evidentes si se omite su análisis.

Por lo tanto, es posible ver que existen áreas de oportunidad si se analizan las partes contrarias a la información que habitualmente tratamos de hallar. De este modo es posible extender este trabajo hacia mejores y peores prácticas, beneficios y perjuicios de aplicarlas, y los riesgos y oportunidades según el contexto en el que estas se apliquen.

Amenazas a la Validez

Entre las principales amenazas a la validez se encuentran la limitada selección de estudios debido a restricciones de acceso en las fuentes de información, así como posibles sesgos de interpretación. Para mitigar dichas amenazas, se utilizaron los enfoques sistemáticos propuestos por Kitchenham et al. (2015) y el enfoque Quasi-Gold Standard de Zhang et al. (2011). Junto con la supervisión de asesores y la retroalimentación de expertos, se contribuyó a mitigar dichos sesgos y mejorar la validez de los resultados.

Conclusión

El objetivo de esta investigación fue identificar y analizar prácticas de ciberseguridad aplicadas en el desarrollo de software y compatibles con los fundamentos de la ingeniería de software, integrando evidencia proveniente de la literatura científica mediante una revisión sistemática de la

literatura. Los resultados muestran que las prácticas más relevantes —gestión de riesgos, modelado de amenazas y diseño de *misuse cases*— se concentran en las etapas tempranas del SDLC, confirmando la importancia de incorporar la seguridad desde el inicio del proceso de desarrollo. Asimismo, se identificó una fuerte presencia de marcos y estándares reconocidos (OWASP, NIST, ISO), así como un creciente uso de herramientas de análisis estático y metodologías ágiles con enfoque DevSecOps.

Este trabajo aporta una sistematización actualizada de prácticas, artefactos, herramientas y frameworks de ciberseguridad en la ingeniería de software, ofreciendo una visión integral que conecta la teoría con la práctica. Sus hallazgos pueden servir como referencia para investigadores, docentes y profesionales interesados en incorporar la ciberseguridad en procesos de desarrollo reales y en programas educativos. Como implicación práctica, los resultados apoyan el diseño de estrategias formativas y metodológicas orientadas a fortalecer la conciencia y competencia en ciberseguridad dentro de equipos de desarrollo. En el ámbito académico, contribuye a consolidar la base de conocimiento sobre ciberseguridad en ingeniería de software, identificando tendencias y vacíos para futuras investigaciones.

Entre las líneas futuras destacan la integración de la ciberseguridad en entornos ágiles, el desarrollo de métricas que midan la madurez de la seguridad durante el desarrollo de proyectos de software, la evaluación empírica del impacto de estas prácticas en proyectos de software reales y la exploración profunda sobre las implicaciones individuales, organizacionales y contextuales al integrar prácticas de ciberseguridad.

Referencias

[1]. Lemos, R. (2023, February). *Cyberattack on fintech firm disrupts derivatives trading globally*. Dark Reading. https://www.darkreading.com/cyberattacks-data-breaches/cyberattack-fintech-firm-disrupts-derivatives-trading

[2]. Vijayan, J. (2023, January). *How noob website hackers can become persistent threats*. Dark Reading. https://www.darkreading.com/cyberattacks-data-breaches/noob-hackers-become-persistent-threats

[3]. Vailshery, L. S. (2022, March). *Share of corporate data stored in the cloud in organizations worldwide from 2015 to 2022*. Statista. https://www.statista.com/statistics/1062879/worldwide-cloud-storage-of-corporate-data/

[4]. Sommerville, I. (2011). *Software engineering* (9th ed.). Addison-Wesley.

[5]. Emami, M. S., Ithnin, N. B., & Ibrahim, O. (2010). Software process engineering: Strengths, weaknesses, opportunities and threats. In Proceedings of the 6th International Conference on Networked Computing (INC2010).

[6]. Straub, J. (2020). Software engineering: The first line of defense for cybersecurity. In *2020 IEEE 11th International Conference on Software Engineering and Service Science (ICSESS)*.

[7]. González, H., Llamas Contreras, R., & Montaño Rivas, O. (2019). When software engineering meets cybersecurity at the classroom. In *2019 7th International Conference in Software Engineering Research and Innovation (CONISOFT)*.

[8]. Khan, R. A., Khan, S. U., Ilyas, M., & Idris, M. Y. (2020). The state of the art on secure software engineering: A systematic mapping study. In *Proceedings of the 24th International Conference on Evaluation and Assessment in Software Engineering (EASE '20)*.

[9]. González, H., Llamas Contreras, R., & Guerra García, C. (2021). Cybersecurity practices at the initial stages of the software engineering process. In *2021 9th International Conference in Software Engineering Research and Innovation (CONISOFT)*.

[10]. Khan, R. A., Khan, S. U., Khan, H. U., & Ilyas, M. (2022a). Systematic literature review on security risks and its practices in secure software development. *IEEE Access, 10*, 5456–5481.

[11]. Khan, R. A., Khan, S. U., & Ilyas, M. (2022b). Exploring security procedures in secure software engineering: A systematic mapping study. In *Proceedings of the 26th International Conference on Evaluation and Assessment in Software Engineering (EASE '22)*.

[12]. Selva-Mora, A., & Quesada-López, C. (2024). Security practices in agile software development: A mapping study. In *Proceedings of the 7th ACM/IEEE International Workshop on Software-intensive Business (IWSiB '24)*.

[13]. Kitchenham, B. A., Budgen, D., & Brereton, P. (2015). *Evidence-based software engineering and systematic reviews*. Chapman & Hall/CRC.

[14]. Zhang, H., Babar, M. A., & Tell, P. (2011). Identifying relevant studies in software engineering. *Information and Software Technology, 53*(6), 625–637.

[15]. Popay, J., Roberts, H., Sowden, A., Petticrew, M., Arai, L., Rodgers, M., Britten, N., Roen, K., & Duffy, S. (2006). *Guidance on the conduct of narrative synthesis in systematic reviews*. ESRC Methods Programme. https://www.lancaster.ac.uk/media/lancaster-university/content-assets/documents/fhm/dhr/chir/NSsynthesisguidanceVersion1-April2006.pdf

[16]. Abiona, O. O., Oladapo, O. J., Modupe, O. T., Oyeniran, O. C., Adewusi, A. O., & Komolafe, A. M. (2024). The emergence and importance of DevSecOps: Integrating and reviewing security practices within the DevOps pipeline. *World Journal of Advanced Engineering Technology and Sciences, 11*(2), 127–133.

[17]. Check Point. (2024). *What is secure SDLC?* https://www.checkpoint.com/cyber-hub/cloud-security/what-is-secure-sdlc/

NOTAS BIOGRÁFICAS

Eduardo Antonio Castillo Garrido es Licenciado en Ingeniería de Software por la Facultad de Estadística e Informática de la Universidad Veracruzana en Xalapa, Veracruz. Sus áreas de interés académico y profesional incluyen la ingeniería de requisitos, la arquitectura de software y el diseño de sistemas resilientes. El presente trabajo se deriva de su proyecto de titulación en dicha institución.

Juan Carlos Pérez Arriaga es Licenciado en Informática por la Facultad de Estadística e Informática de la Universidad Veracruzana. Maestro en Ciencias de la Computación por la Fundación Arturo Rosenblueth. Actualmente es profesor de tiempo completo de la Facultad de Estadística e Informática de la Universidad Veracruzana. Cuenta con el reconocimiento del Programa de Desarrollo Profesional Docente (PRODEP). Su labor investigadora comprende las áreas de construcción de software, seguridad en el desarrollo de software y accesibilidad en el desarrollo de software.

Héctor Xavier Limón Riaño trabaja como docente de tiempo completo en la facultad de Estadística e Informática de la Universidad Veracruzana. Actualmente es miembro del Sistema Nacional de Investigadores. Es Licenciado en Informática, con maestría y Doctorado en Inteligencia Artificial. Sus áreas de investigación de interés son Sistemas Multi-Agente, Minería de datos, Ciberseguridad y Sistemas Distribuidos; contando con diversas publicaciones en estas áreas.

Saúl Domínguez-Isidro es profesor e investigador en la Facultad de Estadística e Informática de la Universidad Veracruzana, México. Sus intereses de investigación se centran en ingeniería de software, particularmente en search-based software testing, algoritmos bioinspirados, pruebas automatizadas de software, DevOps y aplicaciones de inteligencia artificial en el desarrollo de software. Ha participado en diversos proyectos de investigación y colabora activamente en la formación de estudiantes de licenciatura y posgrado en ciencias de la computación e ingeniería de software.