Daena: International Journal of Good Conscience. A8.21(1)1-15. Enero-Junio 2026.
ISSN 1870-557X / ISSN 3065-7814

Vibe Coding y Seguridad: Análisis de Riesgos, Vulnerabilidades y Estrategias de Mitigación en el Desarrollo Asistido por IA

Vibe Coding and Security: Risk Analysis, Vulnerabilities and Mitigation Strategies in AI-Assisted Development

Torres Guerrero, Francisco[1]

Trevino Roman, Sergio Alexander[2]

[1] Docente Investigador, Facultad de Ingenieria Mecanica y Electrica (FIME), Universidad Autonoma de Nuevo Leon (UANL), Monterrey, Nuevo Leon, Mexico. francisco.torresg@uanl.edu.mx

[2] Information Security Risk Manager; Especialista en Gobierno, Riesgo y Cumplimiento (GRC); ISO/IEC 27001:2022 Lead Auditor Professional (BSI); Miembro de ISACA, Monterrey, Nuevo Leon, Mexico. sergio.trevino@uanl.edu.mx

Recibido: enero 2026. Aceptado: febrero 2026. Publicado: marzo 2026.

Resumen El "vibe coding" es un paradigma emergente de desarrollo de software impulsado por inteligencia artificial generativa, acunado por Andrej Karpathy en febrero de 2025, que permite a desarrolladores y no-desarrolladores crear aplicaciones complejas mediante instrucciones en lenguaje natural. Si bien este enfoque democratiza el acceso al desarrollo de software y acelera significativamente la producción de código, introduce una dimensión critica de riesgos de seguridad que requiere atención académica y practica urgente. El presente articulo analiza sistemáticamente las vulnerabilidades de seguridad asociadas al vibe coding, basándose en evidencia empírica de estudios recientes que reportan mas de 2,000 vulnerabilidades identificadas en 5,600 aplicaciones construidas mediante plataformas de vibe coding, y 69 vulnerabilidades detectadas en evaluaciones directas de herramientas lideres como Claude Code, OpenAI Codex, Cursor, Replit y Devin. Se identifican las principales categorías de riesgo: inyección de código, credenciales expuestas, configuraciones inseguras, debilidades en la cadena de suministro de software, y la erosión del conocimiento técnico. Asimismo, se propone un marco de desarrollo responsable asistido por IA que integra principios de seguridad desde la fase de diseño, incluyendo prompts orientados a seguridad, revisión humana obligatoria, integración de herramientas SAST/DAST en pipelines CI/CD, y la formación de una cultura organizacional de seguridad.

Palabras clave. *vibe coding, seguridad de software, inteligencia artificial generativa, vulnerabilidades, LLM, DevSecOps, OWASP.*

Abstract *"Vibe coding" is an emerging software development paradigm driven by generative artificial intelligence, coined by Andrej Karpathy in February 2025, which enables both developers and non-developers to create complex applications through natural language*

instructions. While this approach democratizes access to software development and significantly accelerates code production, it introduces a critical dimension of security risks requiring urgent academic and practical attention. This article systematically analyzes the security vulnerabilities associated with vibe coding, drawing on empirical evidence from recent studies reporting over 2,000 vulnerabilities identified across 5,600 vibe-coded applications, and 69 vulnerabilities detected in direct evaluations of leading tools such as Claude Code, OpenAI Codex, Cursor, Replit, and Devin. Key risk categories are identified: code injection, exposed credentials, insecure configurations, software supply chain weaknesses, and the erosion of technical knowledge. A responsible AI-assisted development framework is proposed, integrating security principles from the design phase, including security-oriented prompting, mandatory human review, SAST/DAST tool integration in CI/CD pipelines, and the cultivation of a security-minded organizational culture.

Key words. *vibe coding, software security, generative artificial intelligence, vulnerabilities, LLM, DevSecOps, OWASP.*

Introducción

El panorama del desarrollo de software esta experimentando una transformación sin precedentes impulsada por la proliferación de modelos de lenguaje de gran escala (LLM, por sus siglas en ingles). En febrero de 2025, el reconocido investigador de inteligencia artificial Andrej Karpathy introdujo el término "vibe coding" para describir un nuevo paradigma donde el desarrollador abandona el control preciso sobre la implementación técnica y, en cambio, se comunica con sistemas de IA en lenguaje natural, aceptando el código generado con mínima o nula revisión critica.

Este fenómeno ha ganado tracción exponencial en la industria tecnológica global. Plataformas como Cursor, Replit, Lovable.dev, GitHub Copilot y Claude Code han democratizado la creación de software de una manera sin precedentes, permitiendo que personas sin formación tecnica profunda puedan desarrollar y desplegar aplicaciones funcionales en cuestión de horas. Si bien los beneficios en términos de productividad son innegables, el vibe coding plantea desafíos fundamentales para la seguridad del software que la comunidad académica y la industria están apenas comenzando a dimensionar.

Los datos son alarmantes. Un estudio del equipo de investigación Escape analizo mas de 5,600 aplicaciones públicamente disponibles construidas con plataformas de vibe coding e identifico más de 2,000 vulnerabilidades de seguridad, más de 400 secretos expuestos y 175 instancias de información de identificacion personal (PII) comprometida (Escape Research Team, 2025). Por su parte, una evaluación comparativa de diciembre de 2025 realizada por la empresa de seguridad Tenzai encontró un total de 69 vulnerabilidades en el código generado por cinco herramientas lideres de vibe coding, incluyendo vulnerabilidades catalogadas como criticas (Tenzai, 2025).

La problemática trasciende el mero debate técnico: cuando personas sin conocimientos de seguridad informática utilizan herramientas de vibe coding para construir aplicaciones que manejan datos sensibles, los riesgos se amplifican dramáticamente. El concepto de Shadow IT de código IA, es decir, aplicaciones construidas fuera de cualquier marco de gobernanza de seguridad, representa una superficie de ataque invisible y en rápida expansión para las organizaciones (Contrast Security, 2025).

El presente articulo realiza un análisis sistemático de los riesgos de seguridad inherentes al vibe coding, examina la evidencia empírica existente sobre vulnerabilidades identificadas, y propone un marco de mejores prácticas para el desarrollo responsable asistido por IA.

Marco Teórico y Conceptual

Definición y Taxonomía del Vibe Coding

El vibe coding puede definirse como un paradigma de desarrollo de software en el que el desarrollador expresa resultados deseados a un sistema de IA, y este actúa como copiloto generando, modificando o refactorizando código en tiempo real, estableciendo un ciclo de retroalimentación entre la intención humana y la generación automatizada. El termino fue acunado por Andrej Karpathy en febrero de 2025 y ha ganado adopción significativa en organizaciones de todos los tamaños, impulsado por la promesa de mayor productividad y la democratización del desarrollo (Karpathy, 2025).

Desde una perspectiva taxonómica, pueden identificarse dos modalidades principales que representan extremos de un espectro continuo. El vibe coding puro se caracteriza por la aceptación acrítica del código generado por IA sin revisión, pruebas exhaustivas ni comprensión profunda de su funcionamiento; prioriza la velocidad sobre la correctitud y la seguridad, y representa el mayor riesgo para entornos productivos. En contraste, el desarrollo responsable asistido por IA utiliza la IA como herramienta amplificadora de productividad, pero el desarrollador conserva la responsabilidad sobre el código generado, aplicando revisión critica, pruebas de seguridad y comprensión del comportamiento del software (Contrast Security, 2025).

El Ecosistema de Herramientas de Vibe Coding

El ecosistema de herramientas de vibe coding ha evolucionado rápidamente. Las plataformas más relevantes se clasifican en tres categorías. Los entornos de desarrollo integrado con IA (AI-first IDEs), como Cursor y Windsurf, representan la nueva generación de entornos donde la IA esta profundamente integrada en el flujo de trabajo. Las plataformas de desarrollo sin código impulsadas por IA, como Lovable.dev, Base44.com y Replit, permiten a usuarios no técnicos construir aplicaciones completas mediante prompts; estas plataformas presentan el mayor riesgo de seguridad dado su público objetivo. Finalmente, los asistentes de código en línea de comandos, como Claude Code, OpenAI Codex y GitHub Copilot, se integran en flujos de trabajo de desarrolladores tradicionales.

Fundamentos de Seguridad en el Desarrollo de Software

El modelo DevSecOps propone integrar la seguridad como elemento transversal en todas las fases del ciclo de vida del desarrollo de software (SDLC), desde el diseño hasta el despliegue y mantenimiento. El Open Web Application Security Project (OWASP) mantiene catálogos de referencia críticos, incluyendo el OWASP Top 10 y el OWASP LLM Top 10 (2025), este último específicamente orientado a identificar riesgos en aplicaciones que integran modelos de lenguaje de gran escala. Los conceptos de Análisis Estático de Código Fuente (SAST), Pruebas de Seguridad de Aplicaciones Dinámicas (DAST) y Análisis de Composición de Software (SCA) conforman el núcleo del arsenal defensivo que el vibe coding tiende a eludir en su búsqueda de velocidad.

Marco Normativo de Referencia

El análisis de riesgos del vibe coding debe situarse dentro de los marcos normativos vigentes. El NIST SP 800-53 Rev. 5 define controles aplicables al ciclo de vida del desarrollo de software, incluyendo el control SA-11 (Developer Testing and Evaluation), que exige evaluaciones de seguridad del software como requisito del proceso de desarrollo (NIST, 2020). En el contexto del vibe coding, este control es directamente vulnerado cuando el código generado se despliega sin validación sistemática.

El marco SOC 2 (AICPA Trust Services Criteria) estructura los controles de seguridad en cinco categorías. El criterio CC6.1 exige controles de acceso lógico efectivos; una lógica de Segregación de Funciones (SoD) mal implementada por un agente de IA constituye una debilidad significativa directamente auditable bajo este criterio (AICPA, 2022). El NIST SP 800-63B establece requisitos para autenticación multifactorial en sistemas que manejan información sensible (NIST, 2017).

En el contexto mexicano, la Ley Federal de Protección de Datos Personales en Posesión de los Particulares (LFPDPPP) establece obligaciones referentes a los derechos ARCO (Acceso, Rectificación, Cancelación y Oposición), así como la obligación de implementar medidas de seguridad razonables. El incumplimiento, frecuente en sitios de vibe coding, expone a los responsables del tratamiento a sanciones del INAI de hasta 320,000 días de salario mínimo.

Taxonomia de Vulnerabilidades en el Vibe Coding

Con base en la revisión de literatura especializada, estudios empíricos y reportes de incidentes documentados, se identifican ocho categorías principales de vulnerabilidades asociadas al vibe coding.

Código Inseguro Generado por el Modelo

Los modelos de lenguaje son entrenados sobre vastos corpus de código públicamente disponible en internet, incluyendo miles de ejemplos de código de baja calidad, desactualizado o con vulnerabilidades conocidas. Un estudio de Veracode (2025) encontró que el 45% del código generado por IA introduce vulnerabilidades de seguridad. El equipo de Red Team de Databricks

documento un caso emblemático: al solicitar la creación de un juego multijugador, el código generado implemento serialización de objetos Python mediante la librería pickle, conocida por permitir ejecución de código arbitrario remoto (Archibald y Kaplan, 2025). El código funcionaba correctamente desde el punto de vista funcional, pero contenía una vulnerabilidad critica.

Inyección de Prompt y Manipulación del Agente

La inyección de prompt ocurre cuando datos maliciosos procesados por el asistente de IA contienen instrucciones ocultas que modifican el comportamiento del modelo. Entre los casos documentados se encuentran: una vulnerabilidad en la interfaz de línea de comandos de Gemini que permitía ejecución arbitraria de comandos desde instrucciones embebidas en un archivo readme.md; y una inyección colocada en un comentario de código fuente que indujo al entorno Windsurf a almacenar instrucciones maliciosas en su memoria a largo plazo durante meses (Kaspersky, 2025). La CVE-2025-55284 en Claude Code ilustra como este vector puede aprovecharse para exfiltrar datos mediante solicitudes DNS sin confirmación del usuario.

Exposición de Credenciales y Secretos

Una de las vulnerabilidades más prevalentes en aplicaciones de vibe coding es la exposición involuntaria de credenciales. El estudio de Escape (2025) sobre 5,600 aplicaciones identifico más de 400 secretos expuestos públicamente, incluyendo tokens de servicio de Supabase recuperables directamente desde paquetes JavaScript del frontend. En el caso de la red social Moltbook, una base de datos Supabase mal configurada expuso 1.5 millones de tokens de autenticación y 35,000 direcciones de correo electrónico directamente en la internet publica, siendo el origen del problema el desarrollo acelerado mediante vibe coding (Wiz Research, 2025).

Configuraciones de Seguridad Insuficientes

Los sistemas de IA generativa tienden a producir configuraciones de seguridad permisivas para facilitar el funcionamiento de la aplicación en entornos de desarrollo. El estudio de Tenzai (2025) documento que, aunque las herramientas evaluadas evitaron vulnerabilidades clásicas como inyección SQL y Cross-Site Scripting (XSS), la seguridad dependiente del contexto especifico de aplicación resulto significativamente más difícil de garantizar.

Riesgos en la Cadena de Suministro de Software

Los asistentes de IA pueden recomendar librerías de terceros que están activamente siendo explotadas o contienen vulnerabilidades conocidas. El riesgo se compone cuando se consideran los modelos envenenados (poisoned models), escenario en que el dataset de entrenamiento ha sido intencionalmente manipulado para generar código con backdoors o vulnerabilidades especificas (TechTarget, 2025).

Servidores MCP Maliciosos y Extensiones Comprometidas

El Model Context Protocol (MCP) ha emergido como estándar para la integración de herramientas con asistentes de IA, pero también como nuevo vector de ataque. La CVE-2025-

54135 (CurXecute) permitió ejecutar comandos arbitrarios en la máquina del desarrollador a través de Cursor. La CVE-2025-53109 (EscapeRoute), detectada en el servidor MCP de Anthropic, permitía lectura y escritura de archivos arbitrarios en el disco del desarrollador (Kaspersky, 2025).

Deuda Técnica y Erosión del Conocimiento

La brecha de comprensión que emerge cuando los desarrolladores no entienden el código generado mediante IA representa un riesgo operacional critico: si el código falla en producción, nadie sabe cómo repararlo. El propio Andrej Karpathy ha advertido sobre este fenómeno, señalando que como nos apoyamos más en IA, nuestro trabajo principal se desplaza de escribir código a revisarlo (Vir, 2026).

Shadow IT de Código IA

Las herramientas de vibe coding han creado un fenómeno nuevo: aplicaciones construidas por usuarios no técnicos de áreas como marketing o recursos humanos, completamente fuera de cualquier marco de gobernanza de TI. Esta tendencia crea una superficie de ataque invisible que los equipos de seguridad no pueden defender porque no tienen visibilidad sobre ella (Contrast Security, 2025).

Evidencia Empírica y Estudios de Caso

Evaluación Comparativa de Herramientas (Tenzai, 2025)

En diciembre de 2025, Tenzai realizo una evaluación sistemática de cinco plataformas lideres de vibe coding, construyendo tres aplicaciones de prueba idénticas con cada herramienta. Los resultados se presentan en la Tabla 1.

Herramienta	Total Vulner.	Criticas	Altas	Bajas/Medias
Claude Code	~4 (menor)	0	4	9
OpenAI Codex	~12	1	2	9
Cursor	~14	2	3	9
Replit	~19	2	8	9
Devin	~20	1	10	9
TOTAL	**69**	**~6**	**~27**	**~45**

Tabla 1. Distribución de vulnerabilidades por herramienta de vibe coding (Tenzai, dic. 2025)

Los resultados revelan que todas las herramientas generaron un numero idéntico de vulnerabilidades bajas a medias (9 cada una), sugiriendo que este nivel es casi inevitable con el estado actual de la tecnología. Notablemente, ninguna herramienta genero vulnerabilidades de inyección SQL o XSS, categorías clásicas del OWASP Top 10, indicando que los modelos modernos han internalizado las mejores prácticas para vulnerabilidades bien documentadas.

Análisis de Aplicaciones en Producción (Escape, 2025)

El análisis de 5,600 aplicaciones de vibe coding realizado por Escape Research Team en octubre de 2025 proporciona la evidencia empírica más amplia disponible. Los hallazgos incluyen: mas de 2,000 vulnerabilidades identificadas en modo pasivo de escaneo; más de 400 secretos expuestos, incluyendo claves de servicio de Supabase recuperables directamente desde paquetes JavaScript del frontend sin autenticación requerida; 175 instancias de PII expuesta, incluyendo registros médicos; y debilidades criticas accesibles directamente mediante endpoints públicos sin requerir acceso privilegiado.

Incidente Moltbook: Caso de Estudio Documental

El incidente de Moltbook en 2025 representa un caso paradigmático de los riesgos del vibe coding en sistemas reales. La empresa de seguridad Wiz descubrió una base de datos Supabase mal configurada que proporciono acceso de lectura y escritura a los datos, exponiendo 1.5 millones de tokens de autenticación, 35,000 direcciones de correo electrónico y mensajes privados entre agentes. La causa raíz no fue un sofisticado ataque cibernético sino una configuración predeterminada de desarrollo nunca revisada antes del despliegue en producción (Wiz Research, 2025).

Escenarios de Riesgo Incremental en el Vibe Coding

Con base en la metodología de modelado de amenazas Microsoft STRIDE y el marco de gestión de riesgos NIST SP 800-30, se identifican tres escenarios de uso creciente de IA generativa que ilustran como el perfil de riesgo escala con la complejidad de la aplicación.

Escenario 1: Aplicación Web de Baja Complejidad

ID	Riesgo	Prob.	Impacto	Control	Estandar
R1.1	XSS en formulario	Alta	Medio	Sanitización de salida, CSP, HttpOnly cookies	OWASP A03, CWE-79
R1.2	Inyeccion SQL	Media	Alto	Prepared statements, ORM	OWASP A03, CWE-89
R1.3	Falta aviso privacidad	Alta	Medio	Privacy notice, consent banner, LFPDPPP	FTC Act, LFPDPPP Art. 15
R1.4	Exposición PII sin cifrado	Media	Alto	HTTPS obligatorio, TLS 1.3	FTC Safeguards Rule
R1.5	Dependencias vulnerables	Alta	Medio	SCA, npm audit, SBOM	NIST SSDF

Tabla 2. Riesgos de seguridad — Escenario 1: Aplicación Web de Baja Complejidad

El primer escenario contempla el uso de vibe coding para construir una página web con formularios de captación de clientes, procesamiento de datos personales y un módulo de chat de soporte. El código resultante frecuentemente presenta: XSS por ausencia de sanitización de salida (CWE-79), inyección SQL si el backend procesa consultas por concatenación de cadenas, y ausencia de Content Security Policy. Adicionalmente, la LFPDPPP exige avisos claros sobre

recopilacion y uso de datos; un formulario sin aviso de privacidad expone a la organización a sanciones regulatorias.

Escenario 2: Sistema con Segregación de Funciones (SoD)

El segundo escenario contempla el desarrollo de un módulo de inventario integrado a un ERP con requisitos de Segregación de Funciones. Los modelos de IA pueden generar matrices RBAC incompletas, lógica condicional que permite a un rol asumir funciones de otro mediante parámetros manipulables, e inconsistencias entre la interfaz de usuario y los endpoints de la API. El marco SOC 2 identifica la ausencia de evidencia de pruebas de SoD como una significant deficiency; el criterio CC6.1 exige controles de acceso lógico efectivos y el CC7.2 requiere audit logging inalterable (AICPA, 2022).

ID	Riesgo	Prob.	Impacto	Control	Criterio SOC 2
R2.1	SoD insuficiente / bypass de roles	Media	Critico	Matriz SoD documentada, pruebas de conflicto	CC6.1, CC7.1
R2.2	API sin validación de permisos	Alta	Alto	Validación server-side en cada endpoint	CC6.1
R2.3	Ausencia de auditoria de transacciones	Alta	Alto	Audit logging inalterable, SIEM	CC7.2
R2.4	Privilegios excesivos por defecto	Media	Medio	Principio de mínimo privilegio	CC6.2
R2.5	Cambios en roles sin aprobacion	Media	Alto	Workflow de cambio con dual approval	CC6.1

Tabla 3. Riesgos de seguridad — Escenario 2: Sistema con SoD (SOC 2)

Escenario 3: Ecosistema de Alta Complejidad

ID	Riesgo	Prob.	Impacto	Control	Marco
R3.1	Secretos expuestos en código/repos	Alta	Critico	Secrets scanning, vault, rotation policy	NIST 800-53 IA-5
R3.2	Cifrado en reposo ausente o debil	Media	Critico	AES-256, gestión de llaves, KMS	NIST 800-53 SC-28
R3.3	MFA no implementado o evadible	Media	Alto	MFA obligatorio, FIDO2/TOTP	NIST 800-63B, IA-2
R3.4	Ciclo de vida del dato no documentado	Alta	Alto	Data mapping, retention policy	NIST 800-60, LFPDPPP
R3.5	Falta de flujos de derechos ARCO/CCPA	Alta	Alto	API de eliminacion, acceso y opt-out funcionales	LFPDPPP Art. 22, CCPA
R3.6	Logs con PII sin protección	Alta	Medio	Redaccion de logs, cifrado, retention policy	NIST 800-53 AU-9
R3.7	Cadena de suministro vulnerable	Media	Alto	SBOM, SCA, firma de artefactos	NIST SSDF, SLSA

Tabla 4. Riesgos de seguridad — Escenario 3: Ecosistema Critico (NIST 800-53 / LFPDPPP)

El tercer escenario contempla un ecosistema integrado en una organización de salud o servicios financieros que incluye aplicación web de clientes, API de interoperabilidad, almacenamiento de datos sensibles y sistemas de autenticación. El ciclo de vida del dato (NIST SP 800-60) debe gestionarse explícitamente en cada fase: creación, almacenamiento, uso, compartición y destrucción. Los modelos de IA generativamente tienden a omitir la fase de destrucción y a implementar cifrado con algoritmos débiles. La gestión de secretos es la vulnerabilidad mas critica en este escenario; el código generado frecuentemente incluye credenciales hardcodeadas y no utiliza vaults seguros (NIST 800-53 IA-5, SC-28).

Marco de Desarrollo Responsable Asistido por IA

La respuesta al problema de seguridad en el vibe coding no consiste en prohibir su uso, sino en establecer un marco de desarrollo responsable que integre la seguridad como elemento constitutivo del proceso creativo asistido por IA. Se propone un marco de seis componentes.

Prompts Orientados a Seguridad

La investigación del equipo de Red Team de Databricks demuestra que el uso de técnicas de prompting especificas puede reducir significativamente la generación de código inseguro. Se identifican tres técnicas fundamentales: el prompt de sistema orientado a seguridad, que incorpora instrucciones de seguridad desde el inicio de la sesión; los prompts específicos del lenguaje, que incluyen instrucciones sobre vulnerabilidades comunes del framework utilizado; y el Chain-of-Thought para seguridad, que solicita al modelo razonar explícitamente sobre implicaciones de seguridad antes de generar código (Archibald y Kaplan, 2025).

Revisión Humana Obligatoria

El código generado por IA debe ser tratado como código de un colaborador junior: ninguna organización permite a un desarrollador sin experiencia desplegar código directamente en producción sin revisión. La revisión humana debe incluir explícitamente: revisión de seguridad de todas las dependencias introducidas; verificación de configuraciones de autenticación y autorización; búsqueda activa de credenciales hardcodeadas; y validación de controles de acceso a nivel de datos.

Integración de Seguridad en el Pipeline CI/CD

Eran Kinsbruner de Checkmarx articula con precisión el desafío: "El mandato de más depuración es el instinto equivocado para un problema de velocidad AI. La única respuesta viable es mover la seguridad al acto de creación" (Dunn, 2026). Este principio se operacionaliza mediante: SAST para análisis automático del código fuente antes del merge; SCA para verificación de dependencias de terceros contra bases de datos CVE; escáner de secretos para detección automática de credenciales; y DAST para pruebas de seguridad sobre la aplicación en ejecución antes del despliegue.

Gobernanza y Control de Acceso de Herramientas IA

Las organizaciones deben establecer políticas claras sobre el uso de herramientas de vibe coding, incluyendo: gestión de plugins como código que requiere revisión y firma; allowlists para extensiones con sandboxing cuando sea técnicamente viable; control de acceso de agentes bajo el principio de mínimo privilegio; y auditoria y trazabilidad que distinga entre código generado por IA y código escrito por humanos.

El IS Risk Manager como Auditor de Vibe Coding

El Information Security Risk Manager se enfrenta a un desafío paradigmático: el código ya no es exclusivamente producto del juicio humano documentable, sino el resultado de una interacción hombre-maquina donde la trazabilidad de las decisiones de seguridad es difusa. Se propone un marco de auditoría de cuatro fases para proyectos que incorporan vibe coding.

1. **Pre-Generación:** Definición de requisitos de seguridad antes de la interacción con el modelo. Inclusión de restricciones explicitas en los prompts.
2. **Post-Generación Inmediata:** Escaneo automático (SAST, SCA, detección de secretos) sobre el código generado antes de que entre al repositorio.
3. **Revisión por Pares Enfocada:** Code review centrado en lógica de negocio, controles de acceso y cumplimiento normativo, incluyendo verificación de SoD y flujos de autenticación.
4. **Validación Pre-Despliegue:** Pruebas de penetración, verificación de controles y generación de evidencia documental para auditorías externas.

El IS Risk Manager debe integrar la auditoria de vibe coding en el gobierno de riesgos corporativo, actualizando registros de riesgos con "código generado por IA no auditado" como amenaza explicita y estableciendo KPIs de cobertura de revisión para proyectos que utilizan LLMs.

Cultura de Seguridad y Educacion Continua

La formación de desarrolladores y usuarios de vibe coding en principios fundamentales de seguridad es indispensable. Esta formación debe cubrir: reconocimiento de patrones comunes de vulnerabilidades; uso responsable de asistentes de IA; comprensión del OWASP Top 10 y el OWASP LLM Top 10; y las responsabilidades legales asociadas al manejo de datos personales bajo la LFPDPPP.

Caso de Estudio: Glamping Manzanos

Descripción del Sitio

Glamping Manzanos es un sitio web comercial construido mediante vibe coding y desplegado en la plataforma Vercel (https://glampingManzanos.vercel.app), correspondiente a un servicio de hospedaje tipo glamping ubicado en la Sierra de Santiago, Nuevo León, México. El

sitio fue desarrollado utilizando Next.js como framework frontend y publicado sin proceso formal de revisión de seguridad, representando un caso real de vibe coding en el contexto del mercado mexicano de turismo.

El sitio incluye páginas de presentación de amenidades, galería fotográfica, información de contacto, ubicación geográfica y un módulo de reservaciones en la ruta /reservar. Para este análisis se realizó una inspección pasiva del sitio, sin pruebas de penetración activas, identificando hallazgos de seguridad visibles desde la perspectiva de un observador externo no autenticado.

Hallazgos de Seguridad Identificados

El análisis pasivo del sitio permitió identificar siete hallazgos de seguridad, clasificados por categoría y nivel de riesgo estimado (ver Tabla 5).

#	Hallazgo	Descripción	Nivel de Riesgo
F1	Dependencia Cross-Origin de Activos	Todas las imágenes se cargan desde glampingManzanos.com/wp-content/uploads/ (WordPress externo). Un compromiso del servidor WordPress permitiría reemplazar imágenes con contenido malicioso.	Medio
F2	Exposición de Datos de Contacto en Texto Plano	Números de teléfono y correo electrónico expuestos sin ofuscación, facilitando captura por bots para spam y spear phishing.	Bajo-Medio
F3	Revelación del Stack Tecnológico	El dominio.vercel.app y las referencias a /wp-content/ permiten perfilar completamente la arquitectura (Next.js + Vercel + WordPress) con mínimo esfuerzo de reconocimiento.	Bajo
F4	Backend WordPress como Superficie de Ataque	WordPress es el CMS mas atacado del mundo. Sin actualizacion continua, representa un vector critico para comprometer el frontend.	Alto
F5	Ausencia de Headers de Seguridad HTTP	No se evidencio configuración de Content Security Policy, X-Frame-Options ni Permissions-Policy, incrementando la superficie de ataques de inyección de contenido.	Medio
F6	Modulo de Reservaciones sin Validación Verificable	Riesgo significativo de: ausencia de validación server-side, falta de protección CSRF, ausencia de rate limiting y manejo inseguro de datos personales de clientes.	Alto (potencial)
F7	Incumplimiento de la LFPDPPP	El sitio recopila datos personales sin presentar un Aviso de Privacidad visible, incumpliendo el articulo 15 de la LFPDPPP. Expone al propietario a sanciones del INAI de hasta 320,000 dias de salario mínimo.	Alto (legal)

Tabla 5. Hallazgos de seguridad en glampingManzanos.vercel.app — análisis pasivo, marzo 2026

Análisis e Implicaciones

El caso de Glamping Manzanos ilustra con precisión el perfil de riesgo típico de un sitio comercial construido mediante vibe coding por un equipo sin formación especializada en seguridad. Los hallazgos no corresponden a vulnerabilidades catastróficas, sino a un patrón sistemático de omisiones de configuración y gobernanza: el sitio funciona, se ve profesional y

cumple su función comercial aparente, pero presenta múltiples vectores de riesgo invisibles para su propietario.

El hallazgo mas critico desde la perspectiva legal (F7) es paradigmático del comportamiento de los modelos de IA: cuando se les pide construir un sitio web para un negocio, optimizan hacia la funcionalidad visible sin incorporar proactivamente los requerimientos legales de compliance que un desarrollador profesional consultaría como parte de su proceso estándar.

Este caso confirma que los riesgos del vibe coding en aplicaciones comerciales reales trascienden las vulnerabilidades en el código ejecutable e incluyen dimensiones de compliance legal, governance de datos personales y arquitectura de dependencias que los sistemas de IA generativa no abordan proactivamente en su estado actual de desarrollo.

Discusión

Los hallazgos de este análisis revelan una tensión fundamental en el paradigma del vibe coding: la misma característica que lo hace revolucionariamente productivo, la abstracción de los detalles de implementación es también la fuente de sus riesgos de seguridad mas profundos. Esta tensión no puede resolverse exclusivamente mediante mejoras técnicas en los modelos de IA, aunque dichas mejoras son parte de la solución.

La evidencia sugiere que los LLMs modernos han internalizado efectivamente los patrones de vulnerabilidades bien documentados (SQLi, XSS), pero siguen siendo incapaces de razonar adecuadamente sobre vulnerabilidades que dependen del contexto especifico de negocio o de las interacciones emergentes entre componentes. Este es precisamente el espacio donde la supervisión humana es más crítica y más difícil de escalar.

El fenómeno del Shadow IT generado por IA merece atención especial. Las herramientas de vibe coding han reducido la barrera de entrada para la creación de software hasta el punto en que cualquier empleado puede convertirse en un desarrollador involuntario, con todas las responsabilidades de seguridad que eso implica pero sin ninguna de la formación necesaria para asumirlas.

Desde la perspectiva de la educación en ingeniería de software, los hallazgos de este estudio tienen implicaciones curriculares importantes. Los programas de formación en ingeniería de sistemas e informática deben incorporar módulos sobre el uso responsable de herramientas de IA para el desarrollo, el reconocimiento y mitigación de vulnerabilidades en código generado por IA, y los marcos legales del desarrollo de software asistido por inteligencia artificial.

Conclusiones

El vibe coding representa una transformación genuinamente disruptiva en el desarrollo de software, con implicaciones profundas tanto para la industria tecnológica como para la educación en ingeniería. La evidencia empírica disponible permite establecer las siguientes conclusiones.

Las herramientas de vibe coding generan vulnerabilidades de seguridad de manera consistente, incluyendo vulnerabilidades críticas, independientemente de la plataforma utilizada. El número de vulnerabilidades bajas a medias es prácticamente universal entre herramientas, mientras que el perfil de vulnerabilidades altas y criticas varia significativamente.

La mitigación efectiva del riesgo requiere una aproximación sistémica y multicapa, que incluya prompts de seguridad, revisión humana, integración de herramientas de seguridad en CI/CD y formación continua. Las técnicas de prompting orientadas a seguridad son efectivas y accesibles, demostrando que incluso usuarios sin experiencia profunda en seguridad pueden mejorar significativamente el perfil de seguridad del código generado.

El fenómeno del Shadow IT de código IA requiere atención urgente de gobernanza organizacional, dado que crea superficies de ataque invisibles. El rol del IS Risk Manager debe evolucionar hacia la auditoria proactiva de artefactos y procesos de vibe coding, utilizando el marco de cuatro fases propuesto en este trabajo.

El incumplimiento normativo es un riesgo subestimado y de alta probabilidad, como lo demuestra el caso de Glamping Manzanos. La ausencia de avisos de privacidad (LFPDPPP), controles de auditoria (SOC 2 CC7.2) y gestión de secretos (NIST 800-53 IA-5) son hallazgos recurrentes que los modelos de IA no resuelven proactivamente. La educación en seguridad del software debe adaptarse urgentemente al paradigma del vibe coding, incorporando en los currículos de ingeniería módulos específicos sobre el uso responsable de herramientas de IA generativa.

Referencias

Archibald, N., y Kaplan, C. (2025, agosto). Passing the Security Vibe Check: The Dangers of Vibe Coding. Databricks Blog. https://www.databricks.com/blog/passing-security-vibe-check-dangers-vibe-coding

American Institute of CPAs [AICPA]. (2022). Trust Services Criteria for Security, Availability, Processing Integrity, Confidentiality, and Privacy. AICPA.

Anthropic. (2025). CVE-2025-55284: Claude Code Agent Data Exfiltration Vulnerability. National Vulnerability Database.

Apiiro. (2025). 4x Velocity, 10x Vulnerabilities: AI Coding Assistants Are Shipping More Risks. https://apiiro.com/blog/4x-velocity-10x-vulnerabilities-ai-coding-assistants-are-shipping-more-risks/

Cámara de Diputados. (2010). Ley Federal de Proteccion de Datos Personales en Posesion de los Particulares. Diario Oficial de la Federacion, 5 de julio de 2010.

Contrast Security. (2025). What is Vibe Coding? Impact, Security Risks, and Vulnerabilities. https://www.contrastsecurity.com/glossary/vibe-coding

CVE. (2025). CVE-2025-54135 (CurXecute): Arbitrary Command Execution in Cursor. National Vulnerability Database.

CVE. (2025). CVE-2025-53109 (EscapeRoute): Arbitrary File Read/Write in Anthropic MCP Server. National Vulnerability Database.

Dunn, J. E. (2026, enero). Output from vibe coding tools prone to critical security flaws, study finds. CSO Online. https://www.csoonline.com/article/4116923/

Escape Research Team. (2025, octubre). Methodology: 2k+ Vulnerabilities in Vibe-Coded Apps. Escape Blog. https://escape.tech/blog/methodology-how-we-discovered-vulnerabilities-apps-built-with-vibe-coding/

Federal Trade Commission [FTC]. (s.f.). FTC Act. 15 U.S.C. § 45. https://www.ftc.gov/legal-library/browse/statutes/federal-trade-commission-act

ICAEW. (2026, febrero). Cyber: the dangers of agents and vibe coding. https://www.icaew.com/insights/viewpoints-on-the-news/2026/feb-2026/cyber-dangers-of-agents-and-vibe-coding

Karpathy, A. (2025, febrero). "Vibe coding" [Post]. X. https://x.com/karpathy

Kaspersky. (2025, octubre). Security risks of vibe coding and LLM assistants for developers. Kaspersky Blog. https://www.kaspersky.com/blog/vibe-coding-2025-risks/54584/

Knostic. (2025). Top Vibe Coding Security Risks and How to Fix Them. https://www.knostic.ai/blog/vibe-coding-security

Microsoft. (s.f.). STRIDE Threat Model. Microsoft Security Documentation. https://docs.microsoft.com/en-us/azure/security/develop/threat-modeling-tool-threats

MITRE. (s.f.). Common Weakness Enumeration (CWE). https://cwe.mitre.org

National Institute of Standards and Technology [NIST]. (2012). SP 800-30 Rev. 1: Guide for Conducting Risk Assessments. NIST.

National Institute of Standards and Technology [NIST]. (2017). SP 800-63B: Digital Identity Guidelines — Authentication and Lifecycle Management. NIST.

National Institute of Standards and Technology [NIST]. (2008). SP 800-60 Vol. 1: Guide for Mapping Types of Information and Information Systems to Security Categories. NIST.

National Institute of Standards and Technology [NIST]. (2020). SP 800-53 Rev. 5: Security and Privacy Controls for Information Systems and Organizations. NIST.

OWASP Foundation. (2024). OWASP Top 10 Web Application Security Risks. https://owasp.org/www-project-top-ten/

OWASP Foundation. (2025). OWASP LLM Top 10. https://owasp.org/www-project-top-10-for-large-language-model-applications/

Tenzai. (2025, diciembre). Comparative Security Evaluation of Vibe Coding Platforms. Reporte tecnico. https://tenzai.io

TechTarget / Smith, M. (2025). Vibe coding security risks and how to mitigate them. SearchSecurity. https://www.techtarget.com/searchsecurity/tip/Vibe-coding-security-risks-and-how-to-mitigate-them

USCS Institute. (2025). What are Vibe Coding Security Risks and How to Eliminate Them? https://www.uscsinstitute.org/cybersecurity-insights/blog/what-are-vibe-coding-security-risks-and-how-to-eliminate-them

Veracode. (2025). GenAI Code Security Report 2025. Veracode Research. https://www.veracode.com

Vir, R. (2026). The Reality of Vibe Coding: AI Agents and the Security Debt Crisis. Towards Data Science. https://towardsdatascience.com/the-reality-of-vibe-coding-ai-agents-and-the-security-debt-crisis/

Wiz Research. (2025). Moltbook Security Disclosure: Misconfigured Database Exposes Agent Social Network Data. Wiz Blog. https://www.wiz.io