2026

Centro de Respuestas a Incidentes Informáticos
CSIRT Académico UNAD

Seguridad en DevSecOps y Desarrollo Seguro de Software

UNA REVISIÓN INTEGRADORA DE FUNDAMENTOS, PRÁCTICAS Y DESAFÍOS EMERGENTES

VIEM
Vicerrectoría de Innovación y Emprendimiento

ECBTI
Escuela de Ciencias Básicas, Tecnología e Ingeniería

Semillero de Investigación
Ceros y Unos

E-boletín Informativo CSIRT Académico UNAD

Medio de divulgación: Correo Electrónico, Sitio Web

Incluye: Noticias, alertas o informes relacionados con la disciplina de la ciberseguridad

Número Cuarenta [40]
Junio de 2026

Universidad Nacional Abierta y a Distancia (UNAD)

Universidad Nacional Abierta y a Distancia
Calle 14 sur No. 14-23 |Bogotá D.C
Correo electrónico:
csirt@unad.edu.co
Página web: https://csirt.unad.edu.co

Estado legal:
Periodicidad: Mensual
ISSN: 2806-0164

Edición electrónica, financiada por la Universidad Nacional Abierta y a Distancia (UNAD)

Vicerrectoría de Innovación y Emprendimiento (VIEM)
Ing. Andrés Ernesto Salinas
Vicerrector

Escuela de Ciencias Básicas Tecnología e Ingeniería (ECBTI)
Ing. Claudio Camilo González Clavijo
Decano

Maestría en Ciberseguridad (ECBTI)
Ing. Sonia Ximena Moreno Molano
Líder de Programa

Especialización en Seguridad Informática
Ing. Cesar Antonio Villamizar
Líder de Programa

Semillero de Investigación Ceros y Unos, adscrito al Grupo de Byte InDesign

Centro de Desarrollo Tecnológico
CSIRT Académico UNAD
Ing. Luis Fernando Zambrano Hernández
Líder CSIRT Académico UNAD

Responsable de la Edición
Ing. Luis Fernando Zambrano Hernandez

Revisó
Adm. Libardo Cárdenas Corral
Analista CSIRT Académico UNAD

El acelerado ritmo de entrega que caracteriza a los modelos de desarrollo ágil y a las canalizaciones de integración y despliegue continuo (CI/CD)[1] ha obligado a las organizaciones a repensar el lugar que ocupa la seguridad dentro del ciclo de vida del software. DevSecOps surge como respuesta a esta tensión, proponiendo la integración temprana, automatizada y continua de controles de seguridad en cada fase del desarrollo, en lugar de tratarlos como una validación tardía previa al despliegue.

Este boletín académico presenta una revisión integradora de la literatura científica reciente (2023-2025) sobre seguridad en DevSecOps y desarrollo seguro de software, organizada en cinco ejes temáticos

[1] Integración Continua y Entrega/Despliegue Continuo

Tabla de contenido

1. Introducción

La transformación digital ha impulsado a las organizaciones a adoptar prácticas de entrega continua que permiten liberar nuevas versiones de software con una frecuencia sin precedentes. Sin embargo, esta velocidad ha expuesto una tensión estructural: los procesos tradicionales de seguridad, diseñados para ciclos de desarrollo largos y secuenciales, resultan incompatibles con la cadencia de integración y despliegue continuo (CI/CD) propia de DevOps. Cuando la seguridad se aborda como una fase final de verificación, el denominado modelo "shift-right"[2], las vulnerabilidades se detectan tarde, su corrección resulta más costosa y el riesgo de incidentes en producción aumenta considerablemente.

DevSecOps propone revertir esta lógica mediante la integración de la seguridad (Sec) en el flujo de trabajo de desarrollo y operaciones (Dev-Ops), promoviendo la automatización de controles, la colaboración entre equipos y la detección temprana de vulnerabilidades, principio conocido como "shift-left security"[3] (Lombardi & Fanton, 2023). No obstante, la literatura reciente coincide en que la adopción de DevSecOps continúa enfrentando obstáculos relevantes la ausencia de prácticas estandarizadas, la dificultad para lograr interoperabilidad entre herramientas y la resistencia organizacional al cambio cultural que exige este paradigma(Mohammed et al., 2025).

El propósito de este boletín es sintetizar, el estado del conocimiento sobre seguridad en DevSecOps y desarrollo seguro de software. El documento se estructura en cinco ejes temáticos que abarcan desde los fundamentos conceptuales de DevSecOps hasta las dimensiones humanas y de gobernanza que condicionan su implementación efectiva. Esta organización permite ofrecer una panorámica que va de lo general, la evolución del paradigma, a lo específico: las prácticas técnicas de aseguramiento del ciclo de vida, la cadena de suministro y la infraestructura cloud-native y culmina en una reflexión sobre los factores organizacionales que determinan el éxito o el fracaso de su adopción.

---

[2] Enfoque en el desarrollo de software y DevOps que consiste en extender las pruebas, la monitorización y la seguridad hacia las fases posteriores al lanzamiento

[3] Enfoque proactivo en el desarrollo de software que traslada las pruebas y medidas de seguridad a las fases iniciales del ciclo de vida (SDLC

Seguridad en DevSecOps y Desarrollo Seguro de Software

Una revisión integradora de fundamentos, prácticas y desafíos emergentes

Autores

Luis Fernando Zambrano Hernandez
Docente Investigador
Líder CSIRT Académico UNAD
Universidad Nacional Abierta y a Distancia
ORCID: 0000-0002-4690-3526

Hernando José Peña Hidalgo
Docente Investigador
CSIRT Académico UNAD
Universidad Nacional Abierta y a Distancia
ORCID: 0000-0002-3477-2645

Sonia Ximena Moreno Molano
Líder Maestría en Ciberseguridad
Universidad Nacional Abierta y a Distancia
ORCID: 0000-0003-0392-1983

Néstor Raúl Cárdenas Corral
Analista CSIRT Académico UNAD
Universidad Nacional Abierta y a Distancia
ORCID: orcid.org/0000-0003-3691-0148

2. Seguridad en DevSecOps

2.1. Fundamentos, evolución y adopción de DevSecOps

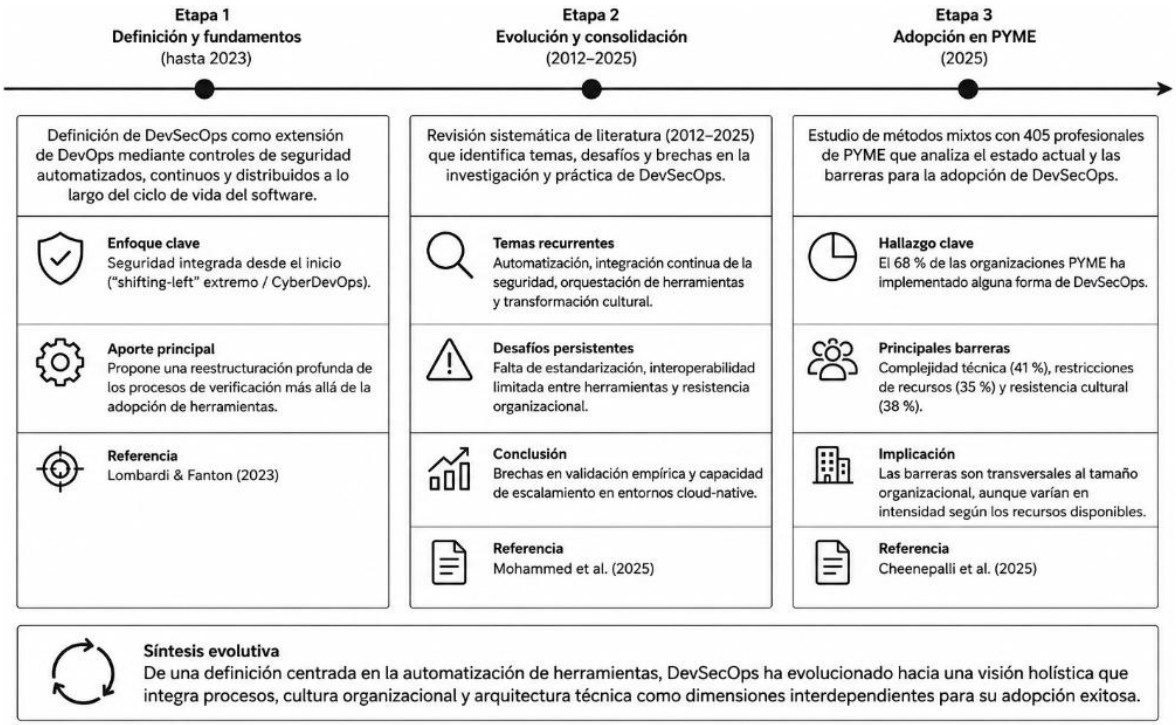
DevSecOps se define como la extensión del paradigma DevOps mediante la incorporación de controles de seguridad automatizados, continuos y distribuidos a lo largo de todas las etapas del ciclo de vida del software, en contraposición a los modelos donde la seguridad se concentra en una auditoría final (Lombardi & Fanton, 2023). Estos autores proponen una arquitectura de "shifting-left" extremo denominada CyberDevOps que integra actividades de ciberseguridad desde las fases más tempranas de la canalización de entrega, argumentando que la sola transición de DevOps a DevSecOps resulta insuficiente si no se acompaña de una reestructuración profunda de los procesos de verificación.

Una revisión sistemática de literatura reciente, que analizó estudios revisados por pares publicados entre 2012 y 2025, identifica como temas recurrentes la automatización, la integración continua de la seguridad, la orquestación de herramientas y la transformación cultural, al tiempo que subraya desafíos persistentes: la ausencia de prácticas estandarizadas, la falta de interoperabilidad entre herramientas de seguridad y la resistencia organizacional (Mohammed et al., 2025). Los

autores concluyen que, pese al creciente interés académico e industrial, todavía existen brechas importantes en la validación empírica de los marcos propuestos y en su capacidad de escalar a entornos cloud-native.

En el ámbito de las pequeñas y medianas empresas (PYME), un estudio de métodos mixtos con 405 profesionales del sector encontró que, si bien el 68 % de las organizaciones ya ha implementado alguna forma de DevSecOps, su adopción se ve limitada por la complejidad técnica (41 % de los encuestados), las restricciones de recursos (35 %) y la resistencia cultural (38 %) (Cheenepalli et al., 2025). Este hallazgo es relevante porque contradice la percepción de que DevSecOps es un dominio exclusivo de grandes corporaciones tecnológicas, evidenciando que las barreras de adopción son transversales al tamaño organizacional, aunque se manifiestan con distinta intensidad según la disponibilidad de recursos especializados.

Ilustración 1.
Evolución y adopción de DevSecOps



Elaboración propia, apoyado con IA

2.2. El ciclo de vida seguro de desarrollo de software (SSDLC) y la seguridad de canalizaciones CI/CD

El Secure Software Development Life Cycle (SSDLC) constituye el marco metodológico que operacionaliza los principios de DevSecOps dentro de las fases clásicas del desarrollo de software: planificación, diseño, implementación, pruebas, despliegue y mantenimiento, incorporando actividades de seguridad específicas en cada una de ellas.

Un análisis comparativo de los ciclos de desarrollo seguros y de los modelos de madurez de seguridad existentes incluyendo el Microsoft SDL, el OWASP Software Assurance Maturity Model (SAMM2) y el DevSecOps Maturity Model (DSOMM), concluye que ninguno de estos marcos históricos fue diseñado originalmente para dar soporte adecuado a soluciones alojadas en la nube (hosted solutions), lo que deja fases críticas como el retiro seguro de un servicio, insuficientemente cubiertas (Lange & Kunz, 2024). El estudio advierte, además, que la mayoría de los marcos no incorpora de forma satisfactoria prácticas ágiles como el trabajo por sprints, lo que genera fricción cuando se intenta aplicarlos en organizaciones pequeñas o con ciclos de entrega muy cortos.

En el plano de la automatización, se ha propuesto un marco unificado para automatizar el análisis de seguridad del software dentro de flujos DevSecOps, orientado a reducir la dependencia de revisiones manuales y a estandarizar la ejecución de análisis estático (SAST) y dinámico (DAST) dentro de la canalización de integración continua (Aljohani & Alqahtani, 2023). Este tipo de propuestas responde a una de las críticas más frecuentes hacia el desarrollo seguro tradicional: la lentitud y variabilidad de las revisiones de seguridad realizadas por personas, que resultan incompatibles con la cadencia de despliegues múltiples por día característica de los entornos DevOps maduros.

Una revisión sistemática de 66 estudios sobre CI/CD para computación en la nube segura documenta el estado actual de las herramientas como: Harbor, SonarQube y GitHub Actions, empleadas para asegurar las distintas etapas de integración, entrega y despliegue continuo, y resalta que buena parte de la investigación sigue concentrada en aspectos técnicos de

herramientas, mientras que persisten vacíos en la evaluación de su efectividad conjunta a escala organizacional (Saleh et al., 2024). Los hallazgos convergen en que la seguridad de la canalización CI/CD no puede entenderse como la suma de controles puntuales, sino como una propiedad emergente de la arquitectura completa del pipeline, que incluye la gestión de secretos, el control de acceso a los sistemas de compilación y la verificación de la integridad de los artefactos generados.

2.3. Seguridad de la cadena de suministro de software: SBOM y análisis de composición (SCA)

La seguridad de la cadena de suministro de software se ha convertido en una prioridad estratégica a raíz de incidentes de alto impacto como el ataque a SolarWinds, que evidenciaron la vulnerabilidad de las dependencias de terceros integradas en el software moderno.

En este contexto, el Software Bill of Materials (SBOM), un inventario formal de los componentes que integran una aplicación, se ha posicionado como una práctica esencial de seguridad, impulsada en Estados Unidos por la Orden Ejecutiva 14028[4]. Un estudio que analizó doscientos artículos y recursos en línea sobre SBOM identificó los cinco beneficios y los cinco desafíos más relevantes de su adopción, concluyendo que, pese al respaldo regulatorio, la generación, mantenimiento y consumo efectivo de los SBOM por parte de las organizaciones todavía dista de ser una práctica madura (Zahan et al., 2023).

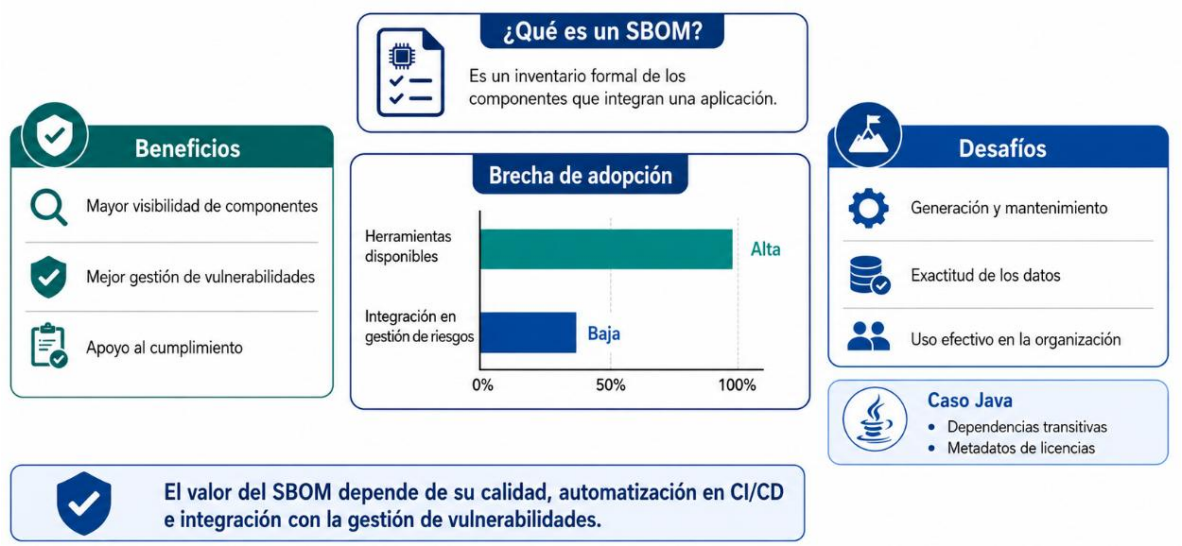
De forma complementaria, un estudio empírico a gran escala sobre el estado de adopción del SBOM en la industria del software examina de qué manera los proyectos de código abierto y comercial están incorporando o no incorporando la generación sistemática de estos inventarios, evidenciando una brecha significativa entre la disponibilidad de herramientas de generación automática (como Syft o CycloneDX) y su integración efectiva dentro de los procesos de gestión de riesgos (Xia et al., 2023). Los autores señalan que persisten problemas de exactitud e integridad en los SBOM generados

[4] https://www.nist.gov/itl/executive-order-14028-improving-nations-cybersecurity

automáticamente, lo que limita su utilidad para la toma de decisiones de seguridad en tiempo real.

En un dominio más específico, un análisis centrado en el ecosistema Java documenta los desafíos particulares que enfrentan los equipos de desarrollo al intentar producir SBOM completos y precisos para aplicaciones basadas en gestores de dependencias como Maven, entre ellos la resolución incompleta de dependencias transitivas y la falta de metadatos de licenciamiento confiables (Balliu et al., 2023). Estos hallazgos refuerzan la idea de que la seguridad de la cadena de suministro no puede resolverse únicamente mediante la adopción de un formato estándar de SBOM, sino que requiere inversión continua en la calidad de los datos, en la automatización de su generación dentro de la canalización CI/CD y en su integración con bases de datos de vulnerabilidades conocidas.

Ilustración 2.
SBOM y seguridad de la cadena de suministro



Elaboración propia apoyada con IA

2.4. Seguridad de contenedores, Kubernetes y arquitecturas cloud-native

La adopción masiva de contenedores y de Kubernetes como plataforma de orquestación ha introducido una superficie de ataque distinta a la de las

arquitecturas monolíticas tradicionales, caracterizada por la naturaleza distribuida, dinámica y efímera de los componentes desplegados.

Una revisión exhaustiva sobre seguridad en servicios cloud-native, sistematiza las principales amenazas (malware, ataques de denegación de servicio distribuido y ataques de intermediario) y las soluciones emergentes, subrayando que la escalabilidad dinámica propia de estos entornos dificulta mantener una postura de seguridad consistente, ya que cada nueva instancia desplegada puede introducir vulnerabilidades no contempladas en el diseño original (Theodoropoulos et al., 2023).

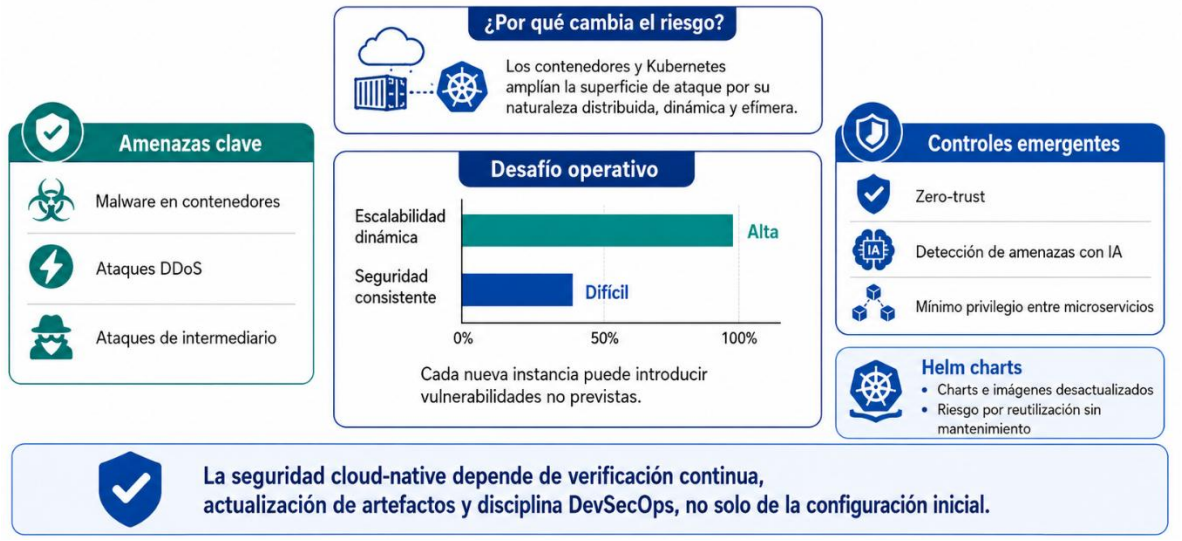
En una línea de investigación más reciente, un estudio sobre técnicas de seguridad centradas en la privacidad y la confianza para entornos cloud-native examina enfoques de arquitectura de confianza cero (zero-trust), detección de amenazas basada en inteligencia artificial y gestión dinámica de la confianza entre microservicios, posicionando estas técnicas como complementarias mas no sustitutas de los controles perimetrales tradicionales (Arif et al., 2025). Los autores destacan que la convergencia entre DevSecOps y arquitecturas cloud-native exige mecanismos de verificación continua de identidad y de autorización de mínimo privilegio entre cargas de trabajo, dado que el perímetro de red deja de ser un límite de confianza fiable.

En el plano operativo, un estudio empírico sobre los Helm charts[5] , analiza su evolución, su grado de desactualización y los riesgos de seguridad asociados a la reutilización de plantillas de despliegue, encontrando que una proporción considerable de las imágenes de contenedor referenciadas por estos charts permanece desactualizada respecto a versiones con vulnerabilidades corregidas (Zerouali et al., 2023). Este hallazgo resulta particularmente relevante para la práctica de DevSecOps, ya que evidencia que la seguridad de la infraestructura como código no depende únicamente de la configuración inicial, sino de la disciplina organizacional para mantener actualizados los

[5] Mecanismo de empaquetado más utilizado para desplegar aplicaciones en Kubernetes (archivos YAML)

artefactos de despliegue a lo largo del tiempo.

Ilustración 3.
Seguridad de contenedores, kubernetes y arquitectura cloud-native



Elaboración propia apoyada con IA

2.5. Factores humanos, gobernanza e inteligencia artificial en la práctica de DevSecOps

Más allá de las herramientas y arquitecturas técnicas, la literatura reciente reconoce que el éxito de DevSecOps depende en gran medida de factores humanos, organizacionales y de gobernanza.

Un estudio que propone la aplicación de herramientas de aprendizaje automático a las etapas de verificación y monitoreo dentro de DevSecOps muestra cómo el análisis automatizado de registros (logs) mediante procesamiento de lenguaje natural puede reducir la carga operativa de los equipos de seguridad y acelerar la detección de anomalías, sin que ello elimine la necesidad de criterio experto humano para interpretar los hallazgos y priorizar la respuesta (Cankar et al., 2023).

En el terreno de la inteligencia artificial generativa, un estudio cualitativo sobre el uso de ChatGPT para tareas de seguridad de software, concluye que, si bien los practicantes valoran positivamente su utilidad para tareas como la detección de vulnerabilidades y la generación de información de seguridad,

existe una brecha notable entre la percepción de utilidad y la fiabilidad práctica de las respuestas generadas, lo que exige mecanismos de verificación humana antes de incorporar estas herramientas en decisiones críticas de seguridad (Kholoosi et al., 2024).

Desde una perspectiva de gobernanza y continuidad, un análisis centrado en la resiliencia de la cadena de suministro de software propone estrategias de gestión de riesgo que trascienden la dimensión netamente técnica, entre ellas la formalización de políticas de gobernanza, la capacitación continua de los equipos de desarrollo y la instauración de mecanismos de transparencia con proveedores externos como condiciones necesarias para sostener en el tiempo cualquier iniciativa de seguridad en el ciclo de vida del software (Akinsola & Akinde, 2024). En conjunto, estos estudios evidencian que la madurez de DevSecOps no puede medirse únicamente por el grado de automatización alcanzado, sino que debe incorporar indicadores de cultura organizacional, capacitación y gobernanza del riesgo.

Ilustración 4

Factores humanos, gobernanza e inteligencia artificial en la madurez de DevSecOps

Nota para la figura: La figura sintetiza los principales hallazgos asociados a la práctica de DevSecOps, destacando que la automatización, el análisis de logs y la inteligencia artificial pueden fortalecer la detección y respuesta ante riesgos, pero requieren verificación humana, políticas de gobernanza, capacitación continua y transparencia con proveedores para garantizar una gestión de seguridad sostenible y confiable.

3. Conclusiones

Este boletín permite identificar convergencias claras en el estado actual del conocimiento sobre seguridad en DevSecOps y desarrollo seguro de software. En primer lugar, existe consenso en que el principio de "shift-left security" como integración temprana y continua de controles de seguridad, constituye el núcleo conceptual de DevSecOps, pero su implementación efectiva exige una reestructuración de procesos que va más allá de la simple incorporación de herramientas automatizadas de escaneo.

En segundo lugar, la seguridad de la cadena de suministro de software, materializada en prácticas como el SBOM y el análisis de composición de software, se consolida como uno de los frentes de investigación e implementación más activos, impulsado tanto por requerimientos regulatorios como por la evidencia de incidentes reales asociados a dependencias comprometidas. No obstante, persisten brechas importantes entre la disponibilidad de estándares y herramientas, y la calidad, completitud y uso operativo real de los inventarios de software generados.

En tercer lugar, la migración hacia arquitecturas cloud-native y la adopción de Kubernetes han desplazado el foco de la seguridad hacia la gestión dinámica de la confianza entre microservicios y hacia la actualización continua de artefactos de despliegue, evidenciando que las prácticas de seguridad perimetral tradicionales resultan insuficientes en estos entornos.

Finalmente, el análisis confirma que los factores humanos y de gobernanza, constituyen, tanto o más que las limitaciones técnicas, el principal obstáculo para alcanzar niveles de madurez avanzados en DevSecOps. La literatura reciente sugiere que el campo se encuentra en una fase de consolidación conceptual, pero todavía requiere mayor validación empírica de los marcos propuestos en contextos organizacionales diversos, particularmente en pequeñas y medianas empresas y en sectores con distintos niveles de exigencia regulatoria.

4. Recomendaciones

A partir de los hallazgos sintetizados en este boletín, se formulan las siguientes recomendaciones dirigidas a equipos de desarrollo, líderes de seguridad y responsables de la toma de decisiones tecnológicas:

- Adoptar el principio de "shift-left security" de manera progresiva, priorizando la automatización de análisis estático (SAST) y de composición de software (SCA) en las primeras etapas de la canalización CI/CD, antes de extender los controles a fases de despliegue y monitoreo en producción.
- Incorporar la generación y el mantenimiento sistemático de un Software Bill of Materials (SBOM) como práctica estándar, integrándolo con bases de datos de vulnerabilidades conocidas para habilitar la gestión proactiva de riesgos de dependencias de terceros.
- Establecer políticas de actualización periódica de imágenes de contenedor y de artefactos de infraestructura como código (por ejemplo, Helm charts), evitando que la desactualización se convierta en una vulnerabilidad silenciosa dentro de arquitecturas cloud-native.
- Complementar los controles perimetrales tradicionales con enfoques de confianza cero (zero-trust) y verificación continua de identidad entre microservicios, especialmente en organizaciones que operan sobre Kubernetes u otras plataformas de orquestación distribuida.
- Invertir en la dimensión cultural y humana de DevSecOps mediante programas de capacitación continua, definición de roles compartidos entre desarrollo, operaciones y seguridad, y mecanismos de gobernanza que formalicen la responsabilidad sobre la seguridad del software a lo largo de todo su ciclo de vida.
- Adoptar herramientas de inteligencia artificial y aprendizaje automático como apoyo y no como sustituto del criterio experto humano en tareas de verificación y monitoreo de seguridad, manteniendo mecanismos de validación ante el riesgo de resultados poco fiables o incompletos.
- Promover, desde la academia y la industria, la validación empírica de los marcos de madurez de DevSecOps en contextos organizacionales diversos, con especial atención a las pequeñas y medianas empresas, donde las restricciones de recursos condicionan de manera particular la adopción de estas prácticas.

5. Referentes Bibliográficos

Akinsola, A., & Akinde, A. (2024). Enhancing Software Supply Chain Resilience: Strategy For Mitigating Software Supply Chain Security Risks And Ensuring Security Continuity In Development Lifecycle. https://doi.org/10.48550/ARXIV.2407.13785

Aljohani, M. A., & Alqahtani, S. S. (2023). A Unified Framework for Automating Software Security Analysis in DevSecOps. 2023 International Conference on Smart Computing and Application (ICSCA), 1-6. https://doi.org/10.1109/ICSCA57840.2023.10087568

Arif, T., Jo, B., & Park, J. H. (2025). A Comprehensive Survey of Privacy-Enhancing and Trust-Centric Cloud-Native Security Techniques Against Cyber Threats. Sensors, 25(8), 2350. https://doi.org/10.3390/s25082350

Balliu, M., Baudry, B., Bobadilla, S., Ekstedt, M., Monperrus, M., Ron, J., Sharma, A., Skoglund, G., Soto-Valero, C., & Wittlinger, M. (2023). Challenges of Producing Software Bill of Materials for Java. IEEE Security & Privacy, 21(6), 12-23. https://doi.org/10.1109/MSEC.2023.3302956

Cankar, M., Petrovic, N., Pita Costa, J., Cernivec, A., Antic, J., Martincic, T., & Stepec, D. (2023). Security in DevSecOps: Applying Tools and Machine Learning to Verification and Monitoring Steps. Companion

of the 2023 ACM/SPEC International Conference on Performance Engineering, 201-205. https://doi.org/10.1145/3578245.3584943

Cheenepalli, J., Hastings, J. D., Ahmed, K. M., & Fenner, C. (2025). Advancing DevSecOps in SMEs: Challenges and Best Practices for Secure CI/CD Pipelines (Versión 1). arXiv. https://doi.org/10.48550/ARXIV.2503.22612

Kholoosi, M. M., Babar, M. A., & Croft, R. (2024). A Qualitative Study on Using ChatGPT for Software Security: Perception vs. Practicality (Versión 1). arXiv. https://doi.org/10.48550/ARXIV.2408.00435

Lange, F., & Kunz, I. (2024). Evolution of secure development lifecycles and maturity models in the context of hosted solutions. Journal of Software: Evolution and Process, 36(12), e2711. https://doi.org/10.1002/smr.2711

Lombardi, F., & Fanton, A. (2023). From DevOps to DevSecOps is not enough. CyberDevOps: An extreme shifting-left architecture to bring cybersecurity within software security lifecycle pipeline. Software Quality Journal, 31(2), 619-654. https://doi.org/10.1007/s11219-023-09619-3

Mohammed, K., Shanmugam, B., & El-Den, J. (2025). Evolution of DevSecOps and Its Influence on Application Security: A Systematic Literature Review. Technologies, 13(12), 548. https://doi.org/10.3390/technologies13120548

Saleh, S., Madhavji, N., & Steinbacher, J. (2024). A Systematic Literature Review on Continuous Integration and Deployment (CI/CD) for Secure Cloud Computing: Proceedings of the 20th International Conference on Web Information Systems and Technologies, 331-341. https://doi.org/10.5220/0013018500003825

Theodoropoulos, T., Rosa, L., Benzaid, C., Gray, P., Marin, E., Makris, A., Cordeiro, L., Diego, F., Sorokin, P., Girolamo, M. D., Barone, P., Taleb, T., & Tserpes, K. (2023). Security in Cloud-Native Services: A Survey. Journal of Cybersecurity and Privacy, 3(4), 758-793. https://doi.org/10.3390/jcp3040034

Xia, B., Bi, T., Xing, Z., Lu, Q., & Zhu, L. (2023). An Empirical Study on Software Bill of Materials: Where We Stand and the Road Ahead. 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), 2630-2642. https://doi.org/10.1109/ICSE48619.2023.00219

Zahan, N., Lin, E., Tamanna, M., Enck, W., & Williams, L. (2023). Software Bills of Materials Are Required. Are We There Yet? IEEE Security & Privacy, 21(2), 82-88. https://doi.org/10.1109/MSEC.2023.3237100

Zerouali, A., Opdebeeck, R., & De Roover, C. (2023). Helm Charts for Kubernetes Applications: Evolution, Outdatedness and Security Risks. 2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR), 523-533. https://doi.org/10.1109/MSR59073.2023.00078