Understanding and Mitigating OWASP A03:2025: A Practical Study of Software Supply Chain Attacks

DRAGO, Macarena; TREBINO, Eric

Universidad Tecnológica Nacional, Facultad Regional La Plata
Laboratorio de Ingeniería en Sistemas de Información, LINES UTN FRLP
Av. 60 s/n° esquina 124, CP 1900, La Plata, Buenos Aires, Argentina.
{mdrago; etrebinofigueroa}@alu.frlp.utn.edu.ar

Abstract.

The increasing reliance on complex software ecosystems has introduced new cybersecurity challenges, particularly within software supply chains. Modern applications depend on third-party libraries, package managers, build systems, and distribution infrastructures, which may pose critical risks if not properly managed. The OWASP Top 10 provides a widely recognized framework for identifying the most relevant threats affecting web applications, and its 2025 edition emphasizes the importance of supply chain security through the category A03:2025 – Software Supply Chain Failures.

This work pursues three main objectives: (i) to analyze the mechanisms and technical characteristics of software supply chain attacks, (ii) to demonstrate, through a proof-of-concept scenario, how these weaknesses can be exploited, and (iii) to evaluate the impact of such attacks on real-world systems. Additionally, a set of mitigation strategies aligned with current cybersecurity best practices is proposed.

The main contribution of this work lies in integrating conceptual analysis, practical demonstration, and mitigation strategies into a unified framework that enables a deeper understanding of the associated risks and provides technical criteria for their effective management.

The results show that the introduction of compromised components into the software development lifecycle represents an increasing risk, highlighting the need for organizations to expand their security strategies beyond application code and consider the broader ecosystem of modern software development.

Keywords: cybersecurity, supply chain, dependencies, OWASP Top 10, vulnerability mitigation.

Comprensión y mitigación de OWASP A03:2025: un estudio práctico de ataques a la cadena de suministro de software

Resumen.

La creciente dependencia de ecosistemas de software complejos ha introducido nuevos retos de ciberseguridad, especialmente en las cadenas de suministro de software. Las aplicaciones modernas dependen de bibliotecas de terceros, gestores de paquetes, sistemas de compilación e infraestructuras de distribución, que pueden representar riesgos críticos si no se gestionan adecuadamente. El OWASP Top 10 ofrece un marco reconocido para identificar las amenazas más relevantes en aplicaciones web, y su edición 2025 subraya la importancia de la seguridad de la cadena de suministro mediante la categoría A03:2025 - Fallos en la cadena de suministro de software.

Este trabajo persigue tres objetivos: (i) analizar el funcionamiento y las características técnicas de los ataques a la cadena de suministro, (ii) demostrar mediante un escenario de prueba de concepto cómo las debilidades pueden ser explotadas, y (iii) evaluar el impacto de estos ataques en sistemas reales. Asimismo, se propone un conjunto de estrategias de mitigación alineadas con las mejores prácticas actuales de ciberseguridad.

La principal contribución consiste en integrar análisis conceptual, demostración práctica y propuestas de mitigación en un marco unificado, que permita comprender en profundidad el riesgo asociado y aportar criterios técnicos para su adecuada gestión.

Los resultados evidencian que la introducción de componentes comprometidos en el ciclo de desarrollo constituye un riesgo creciente, lo que exige que las organizaciones amplíen sus estrategias de seguridad más allá del código y consideren el ecosistema completo del software moderno.

Palabras clave: ciberseguridad, cadena de suministro, dependencias, OWASP Top 10, mitigación de vulnerabilidades.

1 Introducción

1.1 Contexto y relevancia de la ciberseguridad

La importancia de la ciberseguridad ha crecido de manera sostenida debido a la digitalización de servicios, la interconexión de infraestructuras críticas y la adopción de tecnologías como la computación en la nube, el Internet de las Cosas (IoT), la inteligencia artificial y el trabajo remoto. Este escenario ha incrementado significativamente la superficie de ataque, favoreciendo la evolución del cibercrimen a escala global y posicionando a la ciberseguridad como un elemento esencial para garantizar la disponibilidad, integridad y confidencialidad de los sistemas.

1.2 Justificación del problema

El desarrollo de software moderno se basa ampliamente en la reutilización de componentes de terceros, dando lugar a complejas redes de dependencias interconectadas. Estudios empíricos han demostrado que los ecosistemas de software, como npm, están estructurados como redes altamente dependientes, donde una proporción significativa de las aplicaciones incorpora múltiples dependencias (Decan et al., 2019; Chen et al., 2021).

Se ha observado que aproximadamente la mitad de las librerías dependen de otros componentes, lo que evidencia el alto nivel de interconexión en las cadenas de suministro de software (Wang et al., 2024). Esta dependencia masiva incrementa la complejidad y amplía la superficie de ataque, ya que vulnerabilidades en componentes externos pueden propagarse a múltiples sistemas (Zerouali et al., 2022).

En consecuencia, los ataques a la cadena de suministro se han consolidado como una de las amenazas más críticas en la actualidad, afectando tanto a organizaciones como a usuarios finales, y evidenciando la necesidad de adoptar enfoques de seguridad que contemplen todo el ciclo de vida del software y la cadena de suministro.

1.3 Objetivo del trabajo

El presente trabajo se enmarca en una línea de investigación orientada al análisis de vulnerabilidades en entornos de desarrollo de software modernos, con especial énfasis en los riesgos asociados a la cadena de suministro de software.

Esta investigación tiene su origen en un trabajo desarrollado en el marco de la Diplomatura en Ciberseguridad Aplicada, a partir del cual se sentaron las bases teóricas y prácticas que dieron lugar a la presente propuesta. Asimismo, la tesis final de dicha diplomatura se sustentó en los resultados preliminares de esta investigación, consolidando su enfoque y relevancia académica.

En este sentido, el objetivo principal de este estudio es analizar en profundidad la vulnerabilidad A03:2025 – Software Supply Chain Failures, identificando sus causas raíz, mecanismos de explotación y posibles estrategias de mitigación dentro del marco del OWASP Top 10. Para ello, se adopta un enfoque que combina análisis conceptual con validación práctica, permitiendo comprender cómo estas vulnerabilidades pueden ser explotadas en escenarios reales y extender el conocimiento generado,

contribuyendo a la formación en ciberseguridad mediante el análisis de casos reales y la propuesta de estrategias aplicables.

2 Marco teórico

2.1 Fundamentos de seguridad en aplicaciones web

La seguridad de las aplicaciones web comprende el conjunto de prácticas, controles y mecanismos orientados a proteger aplicaciones, servicios y APIs frente a amenazas que afectan su confidencialidad, integridad y disponibilidad.

En el contexto actual, caracterizado por arquitecturas distribuidas y una fuerte dependencia de componentes de terceros, la superficie de ataque se ha expandido significativamente. Las aplicaciones ya no dependen únicamente de su propio código, sino también de librerías externas, servicios, pipelines de integración y herramientas de desarrollo (Williams et al., 2025).

En consecuencia, la seguridad de aplicaciones web trasciende la protección del código fuente, extendiéndose a lo largo de toda la cadena de suministro de software, donde vulnerabilidades en componentes externos pueden impactar directamente en los sistemas finales.

2.2 OWASP Top 10 como estándar de referencia

El OWASP Top 10 constituye un estándar ampliamente reconocido para la identificación y priorización de vulnerabilidades en aplicaciones web. Basado en datos empíricos y contribuciones de la comunidad global, este marco permite clasificar los riesgos más críticos y orientar estrategias de mitigación y prevención. En la Figura 1 se presenta la tabla correspondiente al OWASP Top 10 2025, donde se identifican las principales categorías de riesgo contempladas por este marco (Umar et al., 2024).

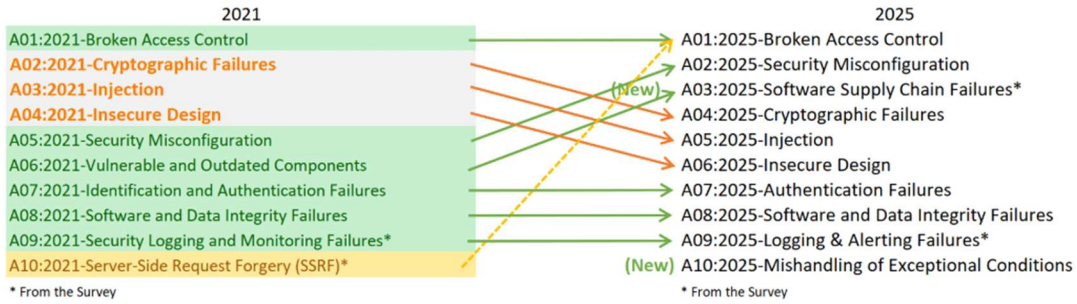

Figura. 1. OWASP Top 10 2025

La edición 2025 refleja una evolución hacia amenazas más complejas, incorporando vulnerabilidades que trascienden el código fuente y afectan al ecosistema completo del desarrollo de software, consolidando su relevancia como guía para la gestión de riesgos en ciberseguridad.

En particular, la categoría A03:2025 – Software Supply Chain Failures representa un cambio significativo en el enfoque de la seguridad de aplicaciones, al ampliar el análisis más allá del código propio e incorporar el ecosistema completo de desarrollo.

2.3 Descripción de la vulnerabilidad A03:2025

La categoría A03:2025 – Software Supply Chain Failures introduce un enfoque centrado en los riesgos asociados a dependencias, herramientas de desarrollo y procesos de distribución. Estas fallas pueden originarse en cualquier etapa del ciclo de vida del software, incluyendo desarrollo, compilación, distribución y actualización (Reichert et al., 2024).

Este tipo de vulnerabilidad se caracteriza por la posibilidad de introducir código malicioso o explotar debilidades en componentes confiables, afectando indirectamente a múltiples sistemas que los incluyen en su cadena de suministro.

2.4 Clasificación y causas comunes

Las vulnerabilidades en la cadena de suministro de software pueden originarse a partir de múltiples factores relacionados con la gestión de dependencias, los procesos de desarrollo y la configuración de los entornos tecnológicos. Estas debilidades, en muchos casos, no responden a errores aislados, sino a deficiencias estructurales en la gobernanza del ciclo de vida del software, lo que incrementa significativamente la superficie de ataque y facilita la explotación por parte de actores maliciosos.

Las principales causas identificadas se resumen en la Tabla 1, donde se describen junto con su impacto potencial en la seguridad de los sistemas.

Tabla. 1. Principales causas de vulnerabilidades en la cadena de suministro de software

Categoría	Descripción	Impacto / Riesgo asociado
Falta de trazabilidad de componentes	Ausencia de mecanismos para identificar origen, versión y relaciones de dependencia.	Dificulta la detección de componentes comprometidos y la respuesta ante incidentes.
Uso de software desactualizado	Utilización de librerías o herramientas con vulnerabilidades conocidas.	Exposición directa a exploits públicos y ataques automatizados.
Ausencia de escaneos de seguridad periódicos	Falta de herramientas automatizadas (SCA, SAST, DAST).	Vulnerabilidades no detectadas en etapas tempranas del desarrollo.
Gestión deficiente de cambios	Falta de controles formales para modificaciones o actualizaciones.	Introducción de vulnerabilidades sin evaluación de riesgos.
Controles de acceso inadecuados	Permisos excesivos o mal configurados en sistemas críticos.	Accesos no autorizados y manipulación maliciosa del código o pipelines.
Dependencias de fuentes no confiables	Uso de paquetes sin verificación de integridad o autenticidad.	Introducción de malware o código comprometido en el sistema.
Demoras en la aplicación de parches	Retrasos en la actualización de componentes vulnerables.	Exposición prolongada a vulnerabilidades conocidas.
Ausencia de pruebas de compatibilidad y seguridad	Falta de validación de nuevas dependencias o actualizaciones.	Fallos funcionales o introducción de nuevas vulnerabilidades.
Configuraciones inseguras	Uso de configuraciones por defecto o incorrectas.	Explotación de configuraciones débiles por atacantes.

Pipelines CI/CD inseguros	Falta de controles en procesos automatizados.	Compromiso del proceso de desarrollo y distribución de software.
Dependencias transitivas no controladas	Inclusión indirecta de librerías sin evaluación de riesgos.	Incremento de la superficie de ataque y propagación de vulnerabilidades.
Falta de monitoreo continuo	Ausencia de detección temprana de anomalías.	Dificultad para identificar ataques en curso o comportamientos sospechosos.

3 Análisis de vulnerabilidades en la cadena de suministro de software

3.1 Funcionamiento y características técnicas

Las vulnerabilidades y exposiciones comunes (CVE) son un conjunto de amenazas de seguridad que se incluyen en un sistema de referencia que describe los riesgos conocidos públicamente. MITRE Corporation, una organización sin fines de lucro que dirige centros federales de investigación y desarrollo patrocinados por gobiernos mantiene la lista de amenazas de CVE.

A pesar de este alcance ampliado, los fallos en la cadena de suministro siguen siendo un reto de identificar. Según la información publicada en el OWASP Top 10, incluida en las referencias del presente trabajo, en el dataset analizado para esta categoría se identificaron 11 CVEs representativos. Asimismo, cuando se analizan los datos aportados, esta categoría presenta la tasa media de incidencia más alta, con un 5,19 %. En la Tabla 2 se detallan las estadísticas correspondientes a esta categoría, incluyendo CWEs mapeados, tasa de incidencia, cobertura, exploit ponderado, impacto ponderado, total de ocurrencias y cantidad total de CVEs. Las CWEs relevantes son:

CWE-477: Uso de funciones obsoletas:

Con la evolución de los lenguajes de programación, algunas funciones dejan de ser recomendadas debido a mejoras en la eficiencia, seguridad y en las convenciones de desarrollo. Estas funciones obsoletas suelen ser reemplazadas por alternativas más modernas que cumplen la misma tarea de manera optimizada y segura, reduciendo riesgos asociados a prácticas antiguas.

CWE-1104: Uso de componentes de terceros no mantenidos:

La dependencia de componentes que han dejado de recibir soporte o actualizaciones representa un riesgo significativo para la seguridad y la calidad del software. La falta de mantenimiento impide aplicar correcciones ante errores o vulnerabilidades conocidas, lo que puede volver obsoleto el código y aumentar la superficie de ataque. Además, esta situación complica las tareas de mantenimiento del producto, prolonga la identificación y mitigación de fallos y, en algunos casos, facilita la introducción de nuevas vulnerabilidades.

CWE-1329: Dependencia en un componente que no es actualizable:

Si se descubre que el componente contiene una vulnerabilidad o un error crítico, pero el problema no se puede solucionar con una actualización o un parche, el propietario del producto no podrá protegerse. La única opción podría ser reemplazar el producto, lo cual podría resultar demasiado costoso financiera u operativamente para el propietario. Como resultado, la imposibilidad de aplicar un parche o una actualización puede dejar el producto vulnerable a la explotación por parte de atacantes o a fallos

operativos críticos. Esta debilidad puede ser especialmente difícil de gestionar al usar ROM, firmware o componentes similares que tradicionalmente han tenido capacidades de actualización limitadas o nulas.

Si bien el hardware puede ser propenso a esta debilidad, los sistemas de software también pueden verse afectados, como cuando un controlador o una biblioteca de terceros ya no reciben mantenimiento o soporte activos, pero siguen siendo fundamentales para la funcionalidad requerida.

CWE-1395: Dependencia de un componente de terceros vulnerable:

Muchos productos son lo suficientemente grandes o complejos como para que parte de su funcionalidad utilice bibliotecas, módulos u otra propiedad intelectual desarrollada por terceros que no son sus creadores. Por ejemplo, en algunos productos de hardware, incluso un sistema operativo completo podría provenir de un proveedor externo. Ya sean de código abierto o cerrado, estos componentes pueden contener vulnerabilidades conocidas públicamente que podrían ser explotadas por adversarios para comprometer el producto.

Tabla. 2. CVEs relacionados a A03:2025 (OWASP)

CWEs Mapeados	5
Tasa máxima de incidencia	8.81%
Tasa media de incidencia	5.19%
Cobertura máxima	65.42%
Cobertura media	28.93%
Exploit ponderado promedio	8.17
Impacto ponderado promedio	5.23
Total de ocurrencias	215,248
Total CVEs	11

3.2 Modelos de ataque en la cadena de suministro

Los ataques a la cadena de suministro pueden clasificarse en distintos modelos según el punto de entrada y el vector utilizado. Entre los más relevantes se encuentran el compromiso de proveedores, la manipulación de dependencias, la alteración de pipelines de integración y despliegue continuo (CI/CD) y la explotación de componentes con vulnerabilidades conocidas.

Estos modelos comparten una característica común: la explotación de relaciones de confianza dentro del ecosistema de desarrollo. En lugar de atacar directamente al sistema objetivo, los atacantes buscan comprometer componentes intermedios que luego serán utilizados por múltiples organizaciones, amplificando el alcance del ataque.

3.3 Mecanismos de explotación conocidos

Los ataques a la cadena de suministro de software se caracterizan por su capacidad de propagarse de manera silenciosa y afectar una gran cantidad de organizaciones mediante la manipulación de componentes que se consideran confiables. A diferencia de las vulnerabilidades tradicionales, donde un atacante compromete directamente un sistema, aplicación o servidor, los ataques de A03:2025 se enfocan en alterar dependencias, herramientas de compilación, repositorios o servicios externos, infiltrándose en el ecosistema antes de llegar al entorno objetivo (Gokkaya et al., 2023). Estos ataques suelen apoyarse en cuatro mecanismos principales:

- Compromiso de proveedores.
- Manipulación de dependencias.
- Alteración de pipelines CI/CD.
- Explotación de componentes con fallas conocidas.

Cada uno de ellos puede habilitar la ejecución de código remoto, robo de credenciales, instalación de backdoor o propagación lateral dentro de entornos de desarrollo y producción.

3.4 Escenarios de ataque

Diversos incidentes reales evidencian la efectividad y el impacto de los ataques a la cadena de suministro de software. Casos ampliamente documentados como SolarWinds, Codecov y Glassworm demuestran cómo la manipulación de componentes considerados confiables puede afectar simultáneamente a miles de organizaciones a nivel global

El caso SolarWinds representa uno de los ejemplos más significativos, donde los atacantes lograron comprometer el proceso de compilación del software Orion, insertando código malicioso en actualizaciones legítimas. Estas actualizaciones fueron distribuidas a aproximadamente 18.000 clientes, incluyendo agencias gubernamentales y grandes corporaciones, lo que permitió a los atacantes obtener acceso persistente a múltiples sistemas críticos (Martínez & Durán, 2021; NPR, 2021).

Por su parte, el incidente de Codecov se originó a partir de la modificación de un script de carga utilizado en su herramienta de cobertura de código. Este script comprometido permitió a los atacantes interceptar y exfiltrar información sensible, como credenciales y tokens, desde los entornos de integración continua (CI) de las organizaciones afectadas, evidenciando la criticidad de asegurar los pipelines de desarrollo (GitGuardian, 2021).

En el caso de Glassworm, se observó una campaña dirigida a desarrolladores mediante la distribución de extensiones maliciosas en el ecosistema de Visual Studio Code. Estas extensiones simulaban ser herramientas legítimas, pero incluían código diseñado para robar información y comprometer entornos de desarrollo, aprovechando la confianza depositada en repositorios y marketplaces oficiales (Fluid Attacks, 2025).

En todos estos escenarios, los atacantes explotaron relaciones de confianza dentro del ecosistema de desarrollo ya sea en proveedores de software, herramientas de desarrollo o repositorios de código para introducir código malicioso que se propagó de manera automática hacia los sistemas de las víctimas. Estos casos ponen de manifiesto que los ataques a la cadena de suministro no requieren comprometer directamente a cada objetivo, sino que pueden escalar mediante la manipulación de un único punto de distribución confiable, amplificando significativamente su alcance e impacto.

4 Prueba de concepto y estudio de caso

4.1 Descripción del escenario

Para validar los conceptos analizados, se diseñó un escenario controlado basado en una organización ficticia, con el objetivo de reproducir un ataque representativo de la vulnerabilidad A03:2025. Este enfoque permitió trasladar el análisis teórico a un contexto práctico, facilitando la comprensión del comportamiento real de este tipo de amenazas.

4.2 Metodología del ataque

El ataque se estructuró en torno a la creación de un paquete malicioso con un nombre similar a una dependencia legítima, técnica conocida como *typosquatting*. Este paquete fue distribuido a través de un repositorio público y complementado con técnicas de ingeniería social para inducir su instalación. En la Figura 2 se puede observar el correo de *phishing* utilizado en la prueba de concepto (POC), mediante el cual se buscó simular el mecanismo de engaño empleado para promover la descarga e instalación del paquete malicioso.

Actualización requerida del módulo de reportes – Acción necesaria

De: soporte-herramienta@capa8-internal.com
Para: equipo.desarrollo@capa8.com
Fecha: 12/11/2025

Estimado equipo,

Se identificó un inconveniente menor en el módulo de generación de reportes de la plataforma interna. Para evitar errores durante la compilación de informes ejecutivos, es necesario aplicar una actualización de dependencias.

Por favor ejecutar el siguiente comando en el entorno virtual del proyecto:

```
pip install reqquests
```

En caso de inconvenientes, responder a este correo o comunicarse con el área de soporte interno.

Atentamente,
Soporte Técnico
Capa Ocho

CAPA 8

Soporte Técnico
Capa Ocho – Seguridad Ofensiva
Mesa de Ayuda Interna
soporte-herramienta@capa8-internal.com
Tel: +54 11 5555-5555

Figura. 2. Correo de phishing utilizado para la introducción del paquete malicioso

4.3 Ejecución de la prueba de concepto

En la prueba de concepto, el código ejecutado durante la instalación se limitó a acciones demostrativas, evidenciando el momento en que el payload es activado. No obstante, en escenarios reales, esta técnica puede ser utilizada para realizar actividades maliciosas como la exfiltración de credenciales, el acceso a variables de entorno sensibles, la descarga de payloads adicionales o el establecimiento de persistencia en el sistema comprometido. Este comportamiento resulta especialmente crítico en entornos automatizados, como pipelines de integración continua (CI/CD), donde la instalación de dependencias se realiza de forma desatendida.

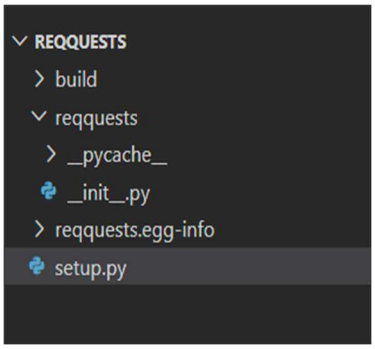

Figura. 3. Estructura del paquete descargado

Figura. 4. Código de instalación de un paquete malicioso implementado en Python mediante el archivo setup.py. Se observa la sobreescritura del comando de instalación (install) a través de una clase personalizada (DemoInstallCommand), lo que permite ejecutar código arbitrario durante el proceso de instalación del paquete.

Figura. 5. Instalación del paquete y ejecución del código sin controles de seguridad

4.4 Resultados obtenidos

Los resultados obtenidos evidenciaron que un ataque de estas características, aplicado en entornos reales, puede comprometer información sensible, incluyendo credenciales, documentos internos y tokens de acceso. Asimismo, permitiría a un atacante establecer persistencia en el sistema y alterar procesos internos, lo que confirma el impacto potencial de las vulnerabilidades asociadas a la cadena de suministro de software.

A partir de la prueba de concepto realizada, se identificó que esta técnica puede aplicarse en distintos escenarios de desarrollo, especialmente en contextos donde se utilizan componentes de terceros, dependencias externas o servicios tercerizados. En este sentido, el análisis refuerza la necesidad de implementar controles específicos sobre la infraestructura, los procesos de desarrollo y la gestión de dependencias, con el fin de fortalecer la seguridad de la cadena de suministro de software.

5 Estrategias de mitigación

La mitigación de vulnerabilidades en la cadena de suministro de software requiere un enfoque integral que contemple tanto aspectos técnicos como organizacionales a lo largo de todo el ciclo de vida del software. En este contexto, resulta fundamental adoptar prácticas que permitan no solo identificar y corregir debilidades existentes, sino también prevenir la introducción de nuevas vulnerabilidades en los procesos de desarrollo y distribución.

La Tabla 3 presenta un conjunto de buenas prácticas alineadas con las recomendaciones de OWASP.

Tabla. 3. Estrategias de mitigación para vulnerabilidades en la cadena de suministro de software

Categoría	**Descripción**	**Medidas clave**
Gestión de dependencias	Consiste en identificar, monitorear y mantener actualizados todos los componentes utilizados por una aplicación a lo largo de su ciclo de vida.	Uso de SBOM para inventario de componentes, control de versiones, identificación de origen y relaciones entre dependencias.
Análisis de dependencia	Incluye la evaluación tanto de dependencias directas como transitivas, las cuales pueden introducir vulnerabilidades de forma indirecta.	Revisión de dependencias transitivas, eliminación de componentes innecesarios, reducción de la superficie de ataque.
Monitoreo de vulnerabilidades	Permite detectar vulnerabilidades en tiempo real mediante herramientas automatizadas que correlacionan dependencias con bases de datos públicas.	Uso de herramientas SCA, consulta de CVE/NVD, detección temprana y priorización de riesgos.
Gestión de fuentes confiables	Garantiza que los componentes provengan de repositorios seguros y verificados.	Descarga desde fuentes confiables, uso de paquetes firmados, validación de integridad.
Gestión de versiones y parches	Implica controlar las actualizaciones de dependencias evitando cambios innecesarios o riesgosos.	Actualizaciones controladas, priorización de parches críticos, reemplazo de componentes obsoletos o sin mantenimiento.
Seguridad en CI/CD	Asegura los procesos automatizados de integración y despliegue, que son puntos críticos en la cadena de suministro.	Autenticación multifactor, control de accesos, separación de funciones, monitoreo continuo.

Protección de activos críticos	Se enfoca en asegurar los elementos clave del entorno de desarrollo y despliegue.	Protección de repositorios, gestión de secretos, firmas digitales, artefactos inmutables, control de versiones.
Gestión de cambios	Permite registrar, auditar y controlar todas las modificaciones en el sistema.	Trazabilidad de cambios, auditorías, control en repositorios, pipelines, configuraciones e integraciones.
Gestión de vulnerabilidades	Establece un proceso continuo para identificar, priorizar y mitigar riesgos a lo largo del ciclo de vida del software.	Supervisión continua, aplicación de parches, gestión proactiva de riesgos.

6 Discusión

Los resultados obtenidos confirman que la cadena de suministro de software constituye uno de los vectores de ataque más relevantes en la actualidad. La creciente dependencia de componentes externos y la complejidad de los ecosistemas de desarrollo incrementan la exposición a este tipo de amenazas.

No obstante, el estudio presenta ciertas limitaciones, principalmente asociadas al uso de un entorno controlado, lo que puede restringir la extrapolación de los resultados a contextos más complejos. Sin embargo, los hallazgos son consistentes con la literatura existente, lo que refuerza la validez del análisis.

7 Conclusiones

7.1 Síntesis de los hallazgos

El presente trabajo ha permitido analizar en profundidad la vulnerabilidad A03:2025 – Software Supply Chain Failures, evidenciando que la creciente dependencia de componentes de terceros y la complejidad de los ecosistemas de desarrollo modernos convierten a la cadena de suministro de software en uno de los principales vectores de ataque en la actualidad. A través de la integración de análisis teórico y validación práctica mediante una prueba de concepto, se ha demostrado cómo debilidades en la gestión de dependencias y en los procesos de desarrollo pueden ser explotadas para comprometer la seguridad de sistemas completos.

Los resultados obtenidos destacan la necesidad de adoptar un enfoque integral de ciberseguridad que trascienda el análisis del código propio e incorpore controles sobre todo el ciclo de vida del software, incluyendo dependencias, herramientas y procesos de integración y despliegue. En este sentido, las estrategias de mitigación propuestas constituyen un conjunto de medidas aplicables que permiten reducir significativamente la superficie de ataque y mejorar la resiliencia de los sistemas.

7.2 Aportes del trabajo

Esta línea de investigación no solo contribuye al entendimiento de la problemática, sino que también establece una base para el desarrollo de trabajos futuros orientados a la aplicación práctica de los mecanismos de mitigación en entornos reales. En particular, se propone avanzar en la implementación de estos controles en sistemas que actualmente no han incorporado prácticas de seguridad en la gestión de la cadena de suministro, con el objetivo de evaluar su efectividad y fortalecer la postura de seguridad organizacional

7.3 Líneas futuras de investigación

En este contexto, se destaca que la institución ha previsto, durante el presente año, la implementación progresiva de los mecanismos de mitigación analizados en este estudio dentro de sus sistemas de información. Esta iniciativa busca mejorar la resiliencia de sus infraestructuras tecnológicas frente a ataques a la cadena de suministro, constituyendo un caso de aplicación concreta que permitirá validar los resultados obtenidos y generar evidencia empírica adicional.

Finalmente, este trabajo sienta las bases para futuras investigaciones orientadas a la automatización de la detección de vulnerabilidades en dependencias, la integración de controles de seguridad en pipelines de desarrollo y la adopción de enfoques proactivos que permitan anticipar y mitigar riesgos en entornos de software cada vez más interconectados y dinámicos.

Durante la preparación de este trabajo, los autores utilizaron ScholarGPT y Perplexity con el fin de mejorar el lenguaje y la legibilidad del manuscrito. Tras utilizar esta herramienta, los autores revisaron y editaron el contenido según fuera necesario y asumen plena responsabilidad por el contenido de la publicación.

Referencias

Chen, X., Abdalkareem, R., Mujahid, S., & Shihab, E. (2021). *Why and how trivial packages impact the npm ecosystem*. Empirical Software Engineering.
https://das.encs.concordia.ca/pdf/Chen_EMSE2021.pdf

Cloudflare. (s. f.). *What is web application security?* https://www.cloudflare.com/es-es/learning/security/what-is-web-application-security/

CrowdStrike. (2025). *Global threat report 2025*.
https://www.crowdstrike.com/en-us/global-threat-report/

Decan, A., Mens, T., & Grosjean, P. (2019). *An empirical comparison of dependency network evolution in seven software packaging ecosystems*. Empirical Software Engineering.
https://arxiv.org/pdf/1710.04936

Fluid Attacks. (2025). *Glassworm: ataque a la cadena de suministro VS Code*. https://fluidattacks.com/es/blog/glassworm-ataque-cadena-de-suministro-vs-code

GitGuardian Blog. (2021). *Codecov supply chain breach*.
https://blog.gitguardian.com/codecov-supply-chain-breach/

Gokkaya, B., Aniello, L., & Halak, B. (2023). *Software supply chain: Review of attacks, risk assessment strategies and security controls*. arXiv.
https://arxiv.org/abs/2305.14157

Martinez Lozano, Jeferson & Durán, Javier. (2021). *Software Supply Chain Attacks, a Threat to Global Cybersecurity: SolarWinds' Case Study. International Journal of Safety and Security Engineering.* 11. 537-545. 10.18280/ijsse.110505.
https://www.researchgate.net/publication/356140449_Software_Supply_Chain_Attacks_a_Threat_to_Global_Cybersecurity_SolarWinds'_Case_Study

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

NPR. (2021). *A worst nightmare cyberattack: The untold story of the SolarWinds hack*. https://www.npr.org/2021/04/16/985439655/a-worst-nightmare-cyberattack-the-untold-story-of-the-solarwinds-hack

OWASP. (2025). *A03:2025 – Software supply chain failures*.
https://owasp.org/Top10/2025/A03_2025-Software_Supply_Chain_Failures/

OWASP. (2025). *Introduction*.
https://owasp.org/Top10/2025/0x00_2025-Introduction/

Reichert, B. M., & Obelheiro, R. R. (2024). Software supply chain security: A systematic literature review. *International Journal of Computers and Applications, 46*(10), 853–867. https://doi.org/10.1080/1206212X.2024.2390978

Umar, R., Riadi, I., & Elfatiha, M. I. A. (2024). Security analysis of web-based academic information system using OWASP framework. *Kinetik, 9*(4).
https://kinetik.umm.ac.id/index.php/kinetik/article/view/2015

Wang, Y., Cheung, S. C., Yu, H., & Zhu, Z. (2024). *Modeling software supply chains and analyzing their evolutionary behaviors*. Springer.
https://www.springerprofessional.de/managing-software-supply-chains/50686510

Williams, L., Benedetti, G., Hamer, S., Paramitha, R., Rahman, I., Tamanna, M., Tystahl, G., Zahan, N., Morrison, P., Acar, Y., Cukier, M., Kästner, C., Kapravelos, A., Wermke, D., & Enck, W. (2025). Research directions in software supply chain security. *ACM Transactions on Software Engineering and Methodology, 34*(5), Article 146. https://doi.org/10.1145/3714464