UNIVERSIDAD MAYOR, REAL Y PONTIFICIA DE SAN FRANCISCO XAVIER DE CHUQUISACA

VICERRECTORADO

CENTRO DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

FACULTAD DE CIENCIAS Y TECNOLOGÍA

ANÁLISIS DEL IMPACTO DE LAS MALAS CONFIGURACIONES EN LA ETAPA DE CI/CD

TRABAJO EN OPCIÓN A DIPLOMADO EN DEVELOPMENT OPERATIONS "DEVOPS" V.1. – MODALIDAD DE GRADUACION

AUTOR: JOSE GAEL CHOQUE SERRANO

SUCRE-BOLIVIA

2024

CESIÓN DE DERECHOS

Al presentar este trabajo como requisito previo para la obtención del Diploma en Developement Operations "DEVOPS" V.1. de la Universidad Mayor, Real y Pontificia de San Francisco Xavier de Chuquisaca, autorizo al Centro de Estudios de Posgrado e Investigación o a la Biblioteca de la Universidad, para que se haga de este trabajo un documento disponible para su lectura según normas de la Universidad

También cedo a la Universidad Mayor, Real y Pontifica de San Francisco Xavier de Chuquisaca, los derechos de publicación de este trabajo o parte de él, manteniendo mis derechos de autor hasta un periodo de 30 meses posterior a su aprobación.

JOSÉ GAEL CHOQUE SERRANO

……..

FIRMA

DEDICATORIA Y AGRADECIMIENTOS

*** A mis queridos padres, **José Luis Choque Velásquez** y **María Yolanda Serrano Martínez**, quienes con su amor incondicional y sabios consejos han sido mi guía y fortaleza en cada paso de mi vida. Su ejemplo de dedicación y esfuerzo ha sido la luz que ha iluminado mi camino.

A mis hermanos, **Cliver Ariel Choque Serrano**, **Yolanda Maylin Choque Serrano** y **Jhossely Mary Choque Serrano**, por ser mi inspiración y mis compañeros de vida. Su apoyo y cariño me han brindado la energía necesaria para alcanzar mis metas.

A mi novia, **Rocío Paola Mendoza Ayaviri**, cuyo amor constante y comprensión han sido un pilar fundamental durante este viaje. Gracias por estar siempre a mi lado, brindándome ánimo y motivación en los momentos más difíciles.

Y a mi pequeño sobrino, **José Matías Quispe Choque**, cuya sonrisa y alegría han llenado de esperanza y felicidad mis días. Tu presencia es un recordatorio de las maravillas que la vida nos ofrece.

A todos ustedes, mi más profundo agradecimiento y amor eterno. Este logro es tan suyo como mío.

INDICE

1 Antecedentes y Justificación 1

2 Situación problemática 2

3 Formulación del problema de investigación 2

4 Objetivo General 2

5 Objetivos específicos 2

6 Diseño Metodológico 3

6.1 Tipo de investigación 3

6.2 Alcance de la Investigación 3

6.3 Métodos 3

6.3.1 Métodos Teóricos 3

6.3.1.1 Método Análisis –Síntesis 3

6.3.1.2 Método revisión de literatura 3

6.3.1.3 Método de la observación 4

6.3.2 Técnicas 4

6.3.2.1 Análisis documental 4

6.3.3 Procedimientos e instrumentos de investigación 4

6.3.3.1 Fichas Técnicas 4

1. MARCO TEORICO Y CONTEXTUAL 5

1.1. Marco Conceptual 5

1.1.1. DevOps 5

1.1.2. Vulnerabilidades 5

1.1.3. Pruebas 5

1.1.4. Despliegue 5

1.1.5. Test 5

1.1.6. Runners 6
1.2. Marco Teórico 6
1.2.1. Introducción al Desarrollo de Software y DevOps 6
1.2.1.1. Evolución del desarrollo de software 6
1.2.1.2. Principios y valores de DevOps 6
1.2.1.3. Fundamentos de DevOps 6
1.2.2. Integración Continua (CI) 8
1.2.2.1. Definición y conceptos clave 8
1.2.2.2. Beneficios de la integración continua 8
1.2.2.3. Herramientas y prácticas comunes de CI 8
1.2.3. Despliegue Continuo (CD) 8
1.2.3.1. Definición y conceptos clave 8
1.2.3.2. Beneficios del despliegue continuo 9
1.2.3.3. Herramientas y prácticas comunes de CD 9
1.2.4. Fundamentos De CI/CD 9
1.2.4.1. Un único repositorio de código fuente 9
1.2.4.2. Registros frecuentes en la sucursal principal 9
1.2.4.3. Compilaciones automatizadas 10
1.2.4.4. Compilaciones de autocomprobación 10
1.2.4.5. Iteraciones frecuentes 10
1.2.4.6. Entornos de prueba estables 10
1.2.4.7. Máxima visibilidad 10
1.2.4.8. Implementaciones predecibles en cualquier momento: 10
1.2.5. Seguridad en DevOps 11
1.2.5.1. Desafíos de seguridad en DevOps 11
1.2.5.2. Impacto de las malas configuraciones en CI/CD en la seguridad 11

1.2.5.3. Mejores prácticas de seguridad en DevOps 11
1.2.5.4. Gestión de identidad y acceso 12
1.2.5.5. Seguridad del código 12
1.2.5.6. Escaneo de vulnerabilidades 12
1.2.5.7. Monitoreo de seguridad 12
1.2.6. Los 10 principales riesgos de seguridad de CI/CD de OWASP 13
1.2.6.1. Introducción 13
1.2.6.2. Definición y antecedentes 13
1.2.6.3. Comprender el riesgo de CI/CD 14
2. DIAGNOSTICO 16
2.1. Introducción 16
2.2. Procesamiento y Análisis de Datos 16
2.3. Tabulación y Codificación de datos 16
2.3.1. Por la naturaleza del riesgo 16
2.3.1.1. Riesgos relacionados con la gestión de identidades y accesos 16
2.3.1.2. Riesgos relacionados con la configuración 16
2.3.1.3. Riesgos relacionados con la integridad de los artefactos 17
2.3.1.4. Riesgos de ejecución de tuberías envenenadas 17
2.3.1.5. Riesgos de control de flujo insuficiente 17
2.3.1.6. Riesgos de higiene insuficiente de las credenciales 17
2.3.1.7. Riesgos de validación incorrecta de la integridad del artefacto 17
2.3.2. Por la fase en el CI/CD 18
2.3.2.1. Riesgos durante la construcción 18
2.3.2.2. Riesgos durante el despliegue 19
2.3.2.3. Riesgos durante el monitoreo 19
2.3.3. Por su impacto 20

2.3.3.1. Riesgos de alto impacto 20
2.3.3.2. Riesgos de impacto moderado 20
2.3.3.3. Riesgos de impacto bajo 20
2.4. Análisis y discusión de resultados 21
2.5. Conclusiones y Recomendaciones 21
2.5.1. Conclusiones 21
2.5.2. Recomendaciones 22
2.5.2.1. Resumen de las recomendaciones de seguridad para entornos de CI/CD: 22
2.5.2.2. Controlar el flujo de código 22
2.5.2.3. Gestionar identidades y accesos 22
2.5.2.4. Evitar el abuso de la cadena de dependencias 22
2.5.2.5. Mitigar la ejecución de pipelines envenenadas (PPE) 23
2.5.2.6. Aplicar controles de acceso basados en pipelines (PBAC) 23
2.5.2.7. Mantener una buena higiene de las credenciales 24
2.5.2.8. Configurar los sistemas de forma segura 24
2.5.2.9. Controlar el uso de servicios de terceros 24
2.5.2.10. Validar la integridad de los artefactos 25
2.5.2.11. Mejorar el registro y la visibilidad 25
Bibliografía 26
ANEXOS 28
3. Ficha Técnica de las configuraciones más comunes con la identificación de sus brechas de seguridad. 28
3.1.1. Mecanismos de control de flujo insuficientes 28
3.1.2. Gestión inadecuada de identidades y accesos 29
3.1.3. Abuso de la cadena de dependencias 30
3.1.4. Ejecución de Tuberías Envenenadas 31

3.1.5. PBAC (controles de acceso basados en tuberías) insuficientes 33
3.1.6. Higiene insuficiente de las credenciales 34
3.1.7. Configuración insegura del sistema 35
3.1.8. Uso no controlado de servicios de terceros 36
3.1.9. Validación incorrecta de la integridad del artefacto 37
3.1.10. Registro y visibilidad insuficientes 38

INDICE DE TABLAS

TABLA 1 MECANISMOS DE CONTROL DE FLUJO 28
TABLA 2 GESTIÓN INADECUADA DE IDENTIDADES Y ACCESOS 29
TABLA 3 ABUSO DE LA CADENA DE DEPENDENCIAS 30
TABLA 4 EJECUCIÓN DE TUBERÍAS ENVENENADAS 31
TABLA 5 PBAC (CONTROLES DE ACCESO BASADOS EN TUBERÍAS) INSUFICIENTES 33
TABLA 6 HIGIENE INSUFICIENTE DE LAS CREDENCIALES 34
TABLA 7 CONFIGURACIÓN INSEGURA DEL SISTEMA 35
TABLA 8 USO NO CONTROLADO DE SERVICIOS DE TERCEROS 36
TABLA 9 VALIDACIÓN INCORRECTA DE LA INTEGRIDAD DEL ARTEFACTO 37
TABLA 10 REGISTRO Y VISIBILIDAD INSUFICIENTES 38

INDICE DE FIGURAS

ILUSTRACIÓN 1 GRAFICA DE RIESGOS SEGÚN SU NATURALEZA 18
ILUSTRACIÓN 2 GRAFICA RIESGOS SEGÚN LA ETAPA 19
ILUSTRACIÓN 3 GRAFICA RIEGOS SEGÚN SU IMPACTO 21

INTRODUCCIÓN

1 Antecedentes y Justificación

La evolución del desarrollo de software ha sido marcada por la constante búsqueda de métodos y prácticas que intentan mejoren la calidad y eficiencia del producto. Es que de esta manera la cultura DevOps surge como un enfoque integral buscando mejorar la comunicación y colaboración de los equipos de desarrollo y operaciones. Sugerida a principios de la década de los años 2000, DevOps se consolida como una cultura por el cual se busca romper las barreras tradicionales entre los equipos de desarrollo y operaciones, fomentando una colaboración más estrecha dando paso a un responsabilidad compartida en la creación y mantenimiento del software (HALL, 2019)

Una de las prácticas clave dentro de la cultura DevOps es la integración continua y el despliegue continuo (CI/CD). La integración continua se refiere a la práctica de fusionar de manera regular los cambios de código en un repositorio compartido, mientras que el despliegue continuo se refiere a la automatización del proceso de entrega de software, desde la compilación hasta la implementación en ambientes de producción. Estas prácticas han demostrado mejorar significativamente la velocidad y la calidad de las entregas de software, al tiempo que reducen los errores y mejoramiento de la calidad.

Sin embargo, a pesar de los beneficios evidentes de DevOps y CI/CD, también existen desafíos significativos. Uno de los desafíos más importantes es la seguridad. Una mala configuración en CI/CD puede dar lugar a vulnerabilidades de seguridad que pueden ser explotadas por actores malintencionados para comprometer la seguridad del desarrollo de software e infraestructura. Estas vulnerabilidades pueden exponer a las organizaciones a riesgos graves, como la pérdida de datos sensibles, el acceso no autorizado y la interrupción de los servicios (Bécares, 2021)

A continuación, mencionaremos uno de los ejemplos más destacado de las consecuencias de una mala configuración en CI/CD. El ataque, que afectó a miles de organizaciones en todo el mundo, expuso la importancia de abordar las brechas de seguridad en la configuración de CI/CD y fortalecer las prácticas de seguridad en el desarrollo de software e infraestructura (Bécares, 2021)

La investigación se justifica por la necesidad de abordar el impacto de las malas

configuraciones en CI/CD en la seguridad del desarrollo de software e infraestructura. Estas configuraciones inseguras pueden exponer a las organizaciones a riesgos graves, como la pérdida de datos sensibles y el acceso no autorizado. Dada la creciente importancia de DevOps en la industria, es crucial comprender y mitigar estos riesgos para garantizar la integridad y seguridad de los sistemas y datos.

Este estudio tiene como objetivo contribuir al conocimiento teórico y metodológico en DevOps y seguridad informática, proporcionando una comprensión más profunda de las vulnerabilidades asociadas con las malas configuraciones en CI/CD.

2 Situación problemática

En el contexto actual de desarrollo de software, la implementación de prácticas de integración continua y despliegue continuo (CI/CD) ha demostrado ser fundamental para mejorar la eficiencia y la velocidad de entrega de software. Sin embargo, las malas configuraciones en CI/CD pueden resultar en vulnerabilidades de seguridad significativas que pueden ser explotadas por actores malintencionados, comprometiendo a las organizaciones a riesgos graves, como la pérdida de datos sensibles, el acceso no autorizado y la interrupción de los servicios.

3 Formulación del problema de investigación

¿Cuál es el impacto de las malas configuraciones en la etapa de CI/CD?

4 Objetivo General

Analizar el impacto que pueden causar las malas configuraciones en la etapa del CI/CD

5 Objetivos específicos

1. Identificar las configuraciones inseguras más comunes en CI/CD.
2. Clasificar el tipo de impacto que tiene cada configuración.
3. Proponer medidas de seguridad para prevenir y mitigar riesgos sobre las brechas de seguridad identificadas.

6 Diseño Metodológico

6.1 Tipo de investigación

La presente investigación es de tipo descriptiva he explicativa, puesto que se busca describir y analizar el impacto de las malas configuraciones en CI/CD "DevOps" en la seguridad del desarrollo de software e infraestructura.

6.2 Alcance de la Investigación

La presente investigación se enfocará en el análisis de las malas configuraciones en CI/CD "DevOps" y su impacto en la seguridad del desarrollo de software e infraestructura.

6.3 Métodos

6.3.1 Métodos Teóricos

6.3.1.1 Método Análisis –Síntesis

Análisis: Este método fue el que más utilizado a lo largo del desarrollo de la monografía, puesto que tiene gran utilidad para la búsqueda y el procesamiento de la información (clasificación, descomposición, división), fue aplicado en las siguientes etapas: en la delimitación del tema, redacción de la situación problemática, la formulación del problema, redacción de los objetivos, así como en el marco teórico.

Síntesis: Este método fue utilizado para la redacción de la introducción, marco teórico, diagnóstico, conclusiones y recomendaciones, puesto que permite la organización y presentación de la información de manera coherente y ordenada.

6.3.1.2 Método revisión de literatura

El método de revisión de literatura es un proceso de búsqueda, selección y análisis de la información relevante sobre un tema específico. Este método fue utilizado para la búsqueda de información sobre las malas configuraciones en CI/CD "DevOps" y su impacto en la seguridad del desarrollo de software e infraestructura este método fue utilizado en la redacción del marco teórico.

6.3.1.3 Método de la observación

Con el método de Observación pudimos observar las practicas comunes de configuración en su entorno natural de trabajo, para identificar las patronas comunes de configuración.

6.3.2 Técnicas

6.3.2.1 Análisis documental

Usamos el método de análisis documental para identificar las configuraciones inseguras más comunes en CI/CD.

6.3.3 Procedimientos e instrumentos de investigación

6.3.3.1 Fichas Técnicas

Utilizamos la información encontrada para la creación de fichas técnicas que nos permitieron identificar las configuraciones inseguras más comunes en CI/CD**.**

CAPITULO I

1. MARCO TEORICO Y CONTEXTUAL

1.1. Marco Conceptual

1.1.1. DevOps

DevOps es una cultura y un conjunto de prácticas que buscan mejorar la colaboración entre los equipos de desarrollo y operaciones, con el objetivo de acelerar la entrega de software y mejorar la calidad del producto final (Humble, 2010).

1.1.2. Vulnerabilidades

Las vulnerabilidades son debilidades en un sistema que pueden ser explotadas por actores malintencionados para comprometer la seguridad del sistema y acceder a información sensible (Mell, 2011).

1.1.3. Pruebas

Las pruebas son un conjunto de actividades que se realizan para verificar el correcto funcionamiento de un sistema o componente de software (Myers, The Art of Software Testing. John Wiley & Sons.).

1.1.4. Despliegue

El despliegue es el proceso de implementar un sistema o componente de software en un entorno de producción (Fowler, 2010).

1.1.5. Test

Los test son un conjunto de actividades que se realizan para verificar el correcto funcionamiento de un sistema o componente de software (Myers, The Art of Software Testing. John Wiley & Sons.).

1.1.6. Runners

Los runners son agentes que ejecutan los pipelines de CI/CD y realizan las tareas de compilación, pruebas y despliegue (Swartout, 2020).

1.2. Marco Teórico

1.2.1. Introducción al Desarrollo de Software y DevOps

1.2.1.1. Evolución del desarrollo de software

El desarrollo de software ha evolucionado significativamente a lo largo de las décadas, desde sus inicios en los años 50 y 60 con metodologías de desarrollo en cascada hasta enfoques más modernos como DevOps. Inicialmente, el desarrollo de software se basaba en modelos secuenciales y lineales, donde cada fase del ciclo de vida del desarrollo (análisis, diseño, implementación, prueba, despliegue) se realizaba de forma independiente y secuencial. Sin embargo, este enfoque presentaba limitaciones en términos de flexibilidad y adaptabilidad a los cambios, lo que dio paso a la adopción de enfoques más ágiles y colaborativos.

1.2.1.2. Principios y valores de DevOps

DevOps es una cultura y un conjunto de prácticas que busca la colaboración y comunicación estrecha entre los equipos de desarrollo de software y operaciones, con el objetivo de acelerar la entrega de software de manera confiable y eficiente. Los principios y valores fundamentales de DevOps incluyen la automatización de procesos, la colaboración entre equipos, la integración continua, la entrega continua, y el monitoreo y retroalimentación constante (Kim & Humble J., 2018) .Estos principios y valores son fundamentales para establecer una cultura DevOps efectiva que permita a las organizaciones desarrollar y desplegar software de manera más rápida y confiable.

1.2.1.3. Fundamentos de DevOps

Cultura de colaboración y responsabilidad compartida La cultura de colaboración en DevOps se basa en la idea de que los equipos de desarrollo y operaciones trabajan juntos de manera más estrecha, compartiendo responsabilidades y objetivos comunes.

Se enfoca en la comunicación abierta, el respeto mutuo y la confianza entre los miembros del equipo (HALL, 2019)

Principales pilares de DevOps Los pilares fundamentales de DevOps incluyen la automatización, la colaboración, la integración continua, la entrega continua, y el monitoreo y retroalimentación constante. Estos pilares son clave para establecer una cultura ágil y eficiente en el desarrollo de software (HALL, 2019)

Automatización La automatización es un aspecto fundamental de DevOps que se centra en la eliminación de tareas manuales repetitivas mediante el uso de herramientas y scripts. La automatización ayuda a reducir errores, mejorar la velocidad y la calidad del desarrollo, y facilitar la entrega continua (Fitzgerald & Stol, 2019).

Colaboración La colaboración en DevOps implica la cooperación estrecha entre los equipos de desarrollo, operaciones y otras áreas relevantes. Se busca eliminar las barreras organizativas y fomentar la comunicación constante para lograr un desarrollo más eficiente y una entrega más rápida de software (Kim, Debois, Willis & Humble, 2016).

Integración continua La integración continua es una práctica en la que los desarrolladores integran su código en un repositorio compartido varias veces al día. Cada integración se verifica automáticamente mediante pruebas para detectar y corregir errores de manera temprana (Fitzgerald & Stol, 2019)).

Entrega continua La entrega continua es una extensión de la integración continua que se enfoca en la automatización del proceso de implementación para que el software pueda entregarse a los usuarios de manera rápida y segura (Fitzgerald & Stol, 2019)

Monitoreo y retroalimentación El monitoreo y la retroalimentación constante son fundamentales en DevOps para garantizar que los sistemas funcionen correctamente y para identificar áreas de mejora. Se utilizan herramientas de monitoreo y análisis para recopilar datos y retroalimentación en tiempo real (Fitzgerald & Stol, 2019)

1.2.2. Integración Continua (CI)

1.2.2.1. Definición y conceptos clave

La Integración Continua (CI) es una práctica de desarrollo de software en la que los miembros de un equipo integran su trabajo con frecuencia, generalmente varias veces al día. Cada integración se verifica mediante la compilación automatizada y las pruebas unitarias, lo que permite detectar y corregir problemas rápidamente (Fitzgerald & Stol, 2019). La CI se centra en la automatización de los procesos de construcción y prueba para mejorar la calidad del software y acelerar su entrega.

1.2.2.2. Beneficios de la integración continua

La CI ofrece varios beneficios, como la detección temprana de errores, la reducción del riesgo de integración y la mejora de la calidad del código (Fitzgerald & Stol, 2019). Al integrar el código de forma continua, los equipos de desarrollo pueden identificar y solucionar problemas de forma proactiva, lo que lleva a un software más estable y confiable.

1.2.2.3. Herramientas y prácticas comunes de CI

Existen diversas herramientas y prácticas comunes utilizadas en la integración continua, como Jenkins, GitLab CI/CD, y Travis CI, que permiten la automatización de los procesos de construcción, prueba e implementación (Fitzgerald & Stol, 2019). Estas herramientas facilitan la colaboración entre los miembros del equipo y garantizan que el código se integre de manera efectiva y eficiente.

1.2.3. Despliegue Continuo (CD)

1.2.3.1. Definición y conceptos clave

El despliegue continuo (CD) es una práctica en DevOps que consiste en automatizar el proceso de implementación de software en entornos de producción después de pasar las pruebas de integración continua (CI) (Fitzgerald & Stol, 2019). CD se enfoca en garantizar que el software pueda ser implementado de manera segura y confiable

en cualquier momento, lo que permite a los equipos de desarrollo entregar cambios de código de forma rápida y frecuente.

1.2.3.2. Beneficios del despliegue continuo

El despliegue continuo ofrece varios beneficios, como la reducción del tiempo de entrega del software, la detección temprana de errores y la mejora de la colaboración entre los equipos de desarrollo y operaciones (Fitzgerald & Stol, 2019). Al automatizar el proceso de implementación, CD permite a los equipos de desarrollo reducir el riesgo asociado con las implementaciones manuales y garantizar una mayor calidad en el software entregado.

1.2.3.3. Herramientas y prácticas comunes de CD

Existen varias herramientas y prácticas comunes utilizadas en el despliegue continuo, como Docker, Kubernetes, y Ansible, que facilitan la automatización de la implementación y configuración de infraestructuras (Fitzgerald & Stol, 2019). Estas herramientas permiten a los equipos de desarrollo crear entornos de desarrollo y producción consistentes, lo que garantiza una implementación exitosa del software en cualquier entorno.

1.2.4. Fundamentos De CI/CD

1.2.4.1. Un único repositorio de código fuente

se debe utilizar la gestión del código fuente para almacenar todos los archivos y scripts necesarios para crear la aplicación. (Ocean, 2020)

1.2.4.2. Registros frecuentes en la sucursal principal

las actualizaciones de código deben mantenerse más pequeñas y realizarse con más frecuencia para garantizar que las integraciones se realicen de la manera más eficiente posible. (Ocean, 2020)

1.2.4.3. Compilaciones automatizadas

la compilación debe automatizarse y ejecutarse a medida que las actualizaciones se insertan en las ramas de la solución de almacenamiento de código fuente. (Ocean, 2020)

1.2.4.4. Compilaciones de autocomprobación

a medida que se automatizan las compilaciones, debe introducirse pasos en los que el resultado de la compilación se pruebe automáticamente para verificar la integridad, la calidad y el cumplimiento de la seguridad (Ocean, 2020)

1.2.4.5. Iteraciones frecuentes

al realizar confirmaciones frecuentes, los conflictos se producen con menos frecuencia. Por lo tanto, las confirmaciones deben mantenerse más pequeñas y realizarse con regularidad. (Ocean, 2020)

1.2.4.6. Entornos de prueba estables

El código debe probarse en un entorno que imite la producción lo más fielmente posible. (Ocean, 2020)

1.2.4.7. Máxima visibilidad

cada desarrollador debe tener acceso a las compilaciones y el código más recientes para comprender y ver los cambios que se han realizado. (Ocean, 2020)

1.2.4.8. Implementaciones predecibles en cualquier momento:

la canalización debe optimizarse para garantizar que las implementaciones se puedan realizar en cualquier momento casi sin riesgo para la estabilidad de la producción. (Ocean, 2020)

1.2.5. Seguridad en DevOps

La seguridad en DevOps es un aspecto crítico debido a la naturaleza ágil y automatizada de los procesos de desarrollo y despliegue. En este contexto, es fundamental abordar los desafíos de seguridad específicos que surgen en entornos DevOps, así como implementar mejores prácticas para garantizar la protección de los sistemas y datos.

1.2.5.1. Desafíos de seguridad en DevOps

La adopción de DevOps plantea varios desafíos de seguridad, como la velocidad de entrega, la falta de visibilidad y control, la complejidad de los entornos de desarrollo y la integración de herramientas y servicios de terceros. Estos desafíos pueden aumentar la superficie de ataque y dificultar la detección y mitigación de amenazas (Fitzgerald & Stol, 2019)**.**

1.2.5.2. Impacto de las malas configuraciones en CI/CD en la seguridad

Las malas configuraciones en los procesos de integración continua y entrega continua (CI/CD) pueden tener un impacto significativo en la seguridad del desarrollo de software e infraestructura. Estas configuraciones incorrectas pueden exponer vulnerabilidades en el código, facilitar el acceso no autorizado y comprometer la integridad y disponibilidad de los sistemas (Bécares, 2021)

1.2.5.3. Mejores prácticas de seguridad en DevOps

Para mitigar los riesgos de seguridad en entornos DevOps, es fundamental implementar mejores prácticas de seguridad. Esto incluye la gestión adecuada de identidad y acceso, la aplicación de controles de seguridad en el código, el escaneo regular de vulnerabilidades y el monitoreo continuo de la seguridad de los sistemas (Fitzgerald & Stol, 2019).

1.2.5.4. Gestión de identidad y acceso

La gestión de identidad y acceso (IAM) es fundamental en DevOps para garantizar que solo las personas autorizadas tengan acceso a los recursos y datos sensibles. Esto se logra mediante la implementación de políticas de acceso basadas en roles y la utilización de herramientas de gestión de identidad y acceso (Fitzgerald & Stol, 2019)

1.2.5.5. Seguridad del código

La seguridad del código es un aspecto crítico en DevOps, ya que cualquier vulnerabilidad en el código puede ser aprovechada por actores malintencionados. Para garantizar la seguridad del código, es importante realizar análisis estáticos y dinámicos, así como implementar buenas prácticas de codificación segura (Bécares, 2021)

1.2.5.6. Escaneo de vulnerabilidades

El escaneo regular de vulnerabilidades es esencial para identificar y corregir posibles brechas de seguridad en el código y en la infraestructura. Herramientas como SonarQube, OWASP Dependency-Check y Nessus pueden ayudar a identificar y mitigar vulnerabilidades de forma proactiva (Bécares, 2021)

1.2.5.7. Monitoreo de seguridad

El monitoreo continuo de la seguridad es fundamental para detectar y responder rápidamente a posibles amenazas. Herramientas como Splunk, ELK Stack y Prometheus pueden proporcionar visibilidad en tiempo real sobre la seguridad de los sistemas y datos (Bécares, 2021)

En esta sección, se presentarán estudios de caso y ejemplos relevantes que ilustran la importancia de abordar las malas configuraciones en CI/CD y la implementación de buenas prácticas de seguridad en entornos DevOps.

1.2.6. Los 10 principales riesgos de seguridad de CI/CD de OWASP

1.2.6.1. Introducción

Las canalizaciones y los procesos de CI/CD facilitan la creación y las implementaciones de software eficientes y repetibles; como tales, ocupan un papel importante en el SDLC moderno. Sin embargo, dada su importancia y popularidad, las canalizaciones de CI/CD también son un objetivo atractivo para los hackers malintencionados, y su seguridad no puede ser ignorada. El objetivo de esta hoja de referencia es proporcionar a los desarrolladores pautas prácticas para reducir el riesgo asociado con estos componentes críticos. Esta hoja de trucos se centrará en asegurar la tubería en sí. Comenzará proporcionando una breve información de fondo antes de continuar con las mejores prácticas de seguridad de CI/CD específicas.

1.2.6.2. Definición y antecedentes

CI/CD se refiere a un conjunto de procesos en gran medida automatizados que se utilizan para crear y entregar software; A menudo se describe como una tubería que consta de una serie de pasos secuenciales y discretos. Por lo general, la canalización comienza cuando el código en desarrollo se inserta en un repositorio y, si todos los pasos se completan correctamente, finaliza con una solución de software creada, probada e implementada en un entorno de producción. CI/CD se puede descomponer en dos partes distintas: integración continua (CI) y entrega continua y/o implementación continua (CD). CI se centra en la automatización de la compilación y las pruebas; La entrega continua se centra en la promoción de este código compilado en un entorno de ensayo o superior y, en general, en la realización de pruebas automatizadas adicionales. Es posible que la entrega continua y la implementación continua no siempre se distingan en las definiciones de CI/CD; sin embargo, según el NIST, la entrega continua requiere que el código se envíe manualmente a producción, mientras que la implementación continua automatiza incluso este paso (OWASP, 2022).

Los pasos exactos de una canalización de CI/CD pueden variar de una organización a otra y de un proyecto a otro; sin embargo, la automatización, y la repetibilidad y agilidad

que aporta, debe ser un enfoque central de cualquier implementación de CI/CD (OWASP, 2022).

1.2.6.3. Comprender el riesgo de CI/CD

Aunque CI/CD aporta muchos beneficios, también aumenta la superficie de ataque de una organización. Las personas, los procesos y la tecnología son necesarios para CI/CD y todos pueden ser vías de ataque; Los repositorios de código, los servidores de automatización como Jenkins, los procedimientos de implementación y los nodos responsables de ejecutar canalizaciones de CI/CD son solo algunos ejemplos de componentes de CI/CD que pueden ser explotados por entidades maliciosas. Además, dado que los pasos de CI/CD se ejecutan con frecuencia utilizando identidades con privilegios elevados, los ataques exitosos contra CI/CD a menudo tienen un alto potencial de daño. Si una organización decide aprovechar los muchos beneficios de CI/CD, también debe asegurarse de invertir los recursos necesarios para protegerlo adecuadamente; las brechas de seguridad de Codecov y SolarWinds son solo dos ejemplos aleccionadores del impacto potencial del compromiso de CI/CD (OWASP, 2022).

Los métodos específicos que utilizan los atacantes para explotar los entornos de CI/CD son diversos; Sin embargo, ciertos riesgos son más prominentes que otros. Aunque no se debe limitar al conocimiento de ellos, comprender los riesgos más destacados para los entornos de CI/CD puede ayudar a las organizaciones a asignar los recursos de seguridad de manera más eficiente. Los 10 principales riesgos de seguridad de CI/CD de OWASP son recursos valiosos para este propósito; el proyecto identifica los siguientes como los 10 principales riesgos de CI/CD:

CICD-SEC-1: Mecanismos de control de flujo insuficientes

CICD-SEC-2: Gestión inadecuada de identidades y accesos

CICD-SEC-3: Abuso de la cadena de dependencias

CICD-SEC-4: Ejecución de Tuberías Envenenadas (PPE)

CICD-SEC-5: PBAC (controles de acceso basados en tuberías) insuficientes

CICD-SEC-6: Higiene insuficiente de las credenciales

CICD-SEC-7: Configuración insegura del sistema

CICD-SEC-8: Uso no controlado de servicios de terceros

CICD-SEC-9: Validación incorrecta de la integridad del artefacto

CICD-SEC-10: Registro y visibilidad insuficientes

El resto de esta hoja de referencia se centrará en proporcionar orientación para mitigar estos 10 riesgos principales y otros riesgos de CI/CD (OWASP, 2022).

CAPITULO II

2. DIAGNOSTICO

2.1. Introducción

En la elaboración del diagnóstico, se utilizó la metodología de redacción de fichas técnicas debido a la naturaleza técnica de nuestra investigación. Esta elección se debió a la limitada disponibilidad de información, dada la sensibilidad de los datos involucrados. La redacción de fichas técnicas fue propuesta en nuestros objetivos como un método adecuado para abordar esta limitación.

2.2. Procesamiento y Análisis de Datos

Se sistematizaron todos los datos recopilados mediante la aplicación de técnicas de análisis documental, así como el uso de fichas técnicas como instrumento de recopilación y organización de la información. Consultar **ANEXOS**

2.3. Tabulación y Codificación de datos

2.3.1. Por la naturaleza del riesgo

2.3.1.1. Riesgos relacionados con la gestión de identidades y accesos

Estos riesgos se refieren a las vulnerabilidades que pueden surgir debido a una gestión inadecuada de identidades y accesos en el proceso de CI/CD. Esto incluye la falta de autenticación adecuada, autorización inadecuada o falta de control de acceso, lo que podría llevar a accesos no autorizados a sistemas o datos sensibles durante el proceso de CI/CD.

2.3.1.2. Riesgos relacionados con la configuración

Estos riesgos se refieren a las vulnerabilidades derivadas de una configuración incorrecta o insegura de los sistemas, herramientas o entornos utilizados en el proceso de CI/CD. Esto incluye la falta de configuraciones seguras, configuraciones por defecto

no modificadas, o la exposición de información sensible debido a una configuración inadecuada.

2.3.1.3. Riesgos relacionados con la integridad de los artefactos

Estos riesgos se refieren a las vulnerabilidades que pueden comprometer la integridad de los artefactos generados durante el proceso de CI/CD. Esto incluye la posibilidad de que los artefactos sean modificados de manera no autorizada, corrompidos o manipulados, lo que podría llevar a la distribución de software malicioso o defectuoso.

2.3.1.4. Riesgos de ejecución de tuberías envenenadas

Estos riesgos se refieren a las amenazas asociadas con la ejecución de tuberías (pipelines) comprometidas o maliciosas en el proceso de CI/CD. Esto incluye la posibilidad de ejecutar código malicioso durante el proceso de construcción, pruebas o despliegue, lo que podría comprometer la seguridad de todo el sistema.

2.3.1.5. Riesgos de control de flujo insuficiente

Estos riesgos se refieren a la falta de controles adecuados para gestionar y supervisar el flujo de trabajo en el proceso de CI/CD. Esto incluye la falta de validaciones o autorizaciones en puntos críticos del proceso, lo que podría permitir a un atacante manipular o interrumpir el flujo normal de trabajo.

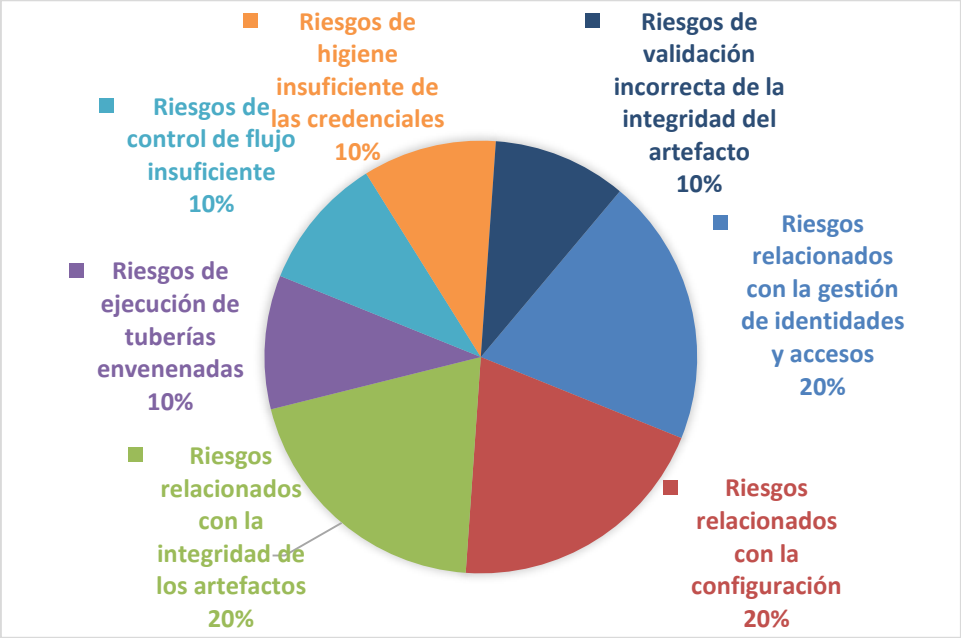
2.3.1.6. Riesgos de higiene insuficiente de las credenciales

Estos riesgos se refieren a la falta de medidas adecuadas para proteger y gestionar las credenciales utilizadas en el proceso de CI/CD. Esto incluye el almacenamiento inseguro de credenciales, la transmisión no cifrada de las mismas o su exposición accidental, lo que podría permitir a un atacante obtener acceso no autorizado a sistemas o servicios.

2.3.1.7. Riesgos de validación incorrecta de la integridad del artefacto

Estos riesgos se refieren a la falta de validaciones adecuadas para garantizar la integridad de los artefactos generados durante el proceso de CI/CD. Esto incluye la falta de verificación de firmas digitales, hash incorrecto o la ausencia de validaciones de integridad, lo que podría permitir la distribución de artefactos modificados o maliciosos.

Ilustración 1 Grafica de riesgos según su naturaleza



Nota: Fuente Propia

2.3.2. Por la fase en el CI/CD

Por supuesto, aquí tienes una descripción más detallada de los riesgos según la fase del CI/CD:

2.3.2.1. Riesgos durante la construcción

Estos riesgos se refieren a las vulnerabilidades que pueden surgir durante la fase de construcción del proceso de CI/CD. Esto incluye la posibilidad de introducir código malicioso o defectuoso durante la compilación, pruebas unitarias y generación de artefactos, lo que podría comprometer la integridad y seguridad del software.

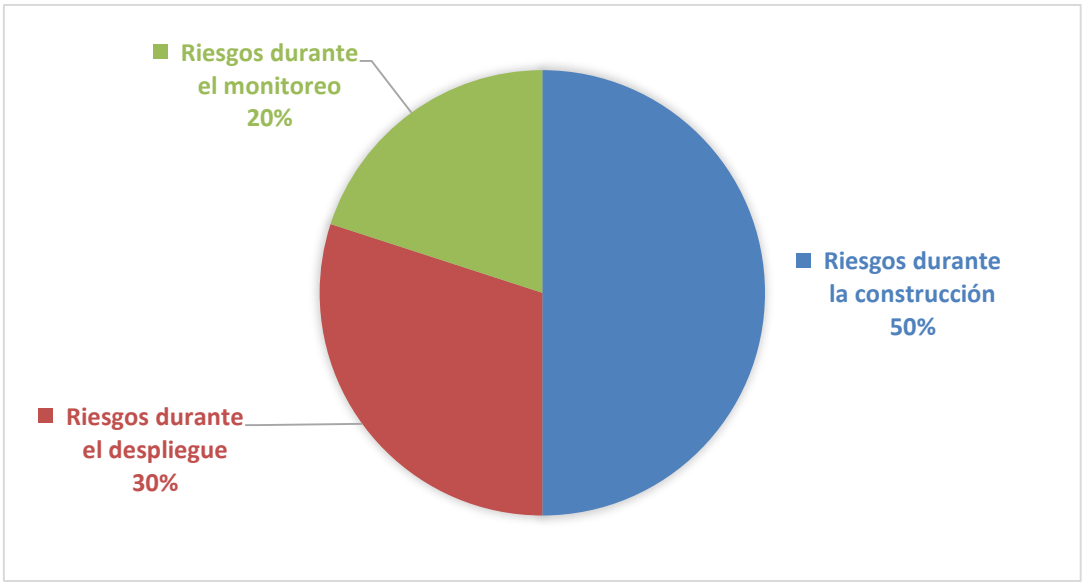
2.3.2.2. Riesgos durante el despliegue

Estos riesgos se refieren a las amenazas asociadas con la fase de despliegue del proceso de CI/CD. Esto incluye la posibilidad de desplegar artefactos vulnerables o maliciosos en entornos de producción, lo que podría resultar en la exposición de sistemas críticos a ataques o la interrupción del servicio.

2.3.2.3. Riesgos durante el monitoreo

Estos riesgos se refieren a las vulnerabilidades que pueden surgir durante la fase de monitoreo y operación del software desplegado. Esto incluye la falta de monitoreo adecuado de los sistemas en producción, la detección tardía de problemas de seguridad o el uso de configuraciones inseguras en entornos de producción, lo que podría facilitar ataques y comprometer la disponibilidad y seguridad del sistema.

Ilustración 2 Grafica riesgos según la etapa



Nota: Fuente Propia

2.3.3. Por su impacto

2.3.3.1. Riesgos de alto impacto

Estos riesgos tienen el potencial de causar daños significativos y generalizados al proceso de CI/CD y al sistema en general. Pueden resultar en la exposición grave de datos sensibles, la interrupción prolongada o la indisponibilidad total del servicio, la pérdida o corrupción masiva de datos, o la compro misión crítica de la integridad y la confidencialidad del sistema.

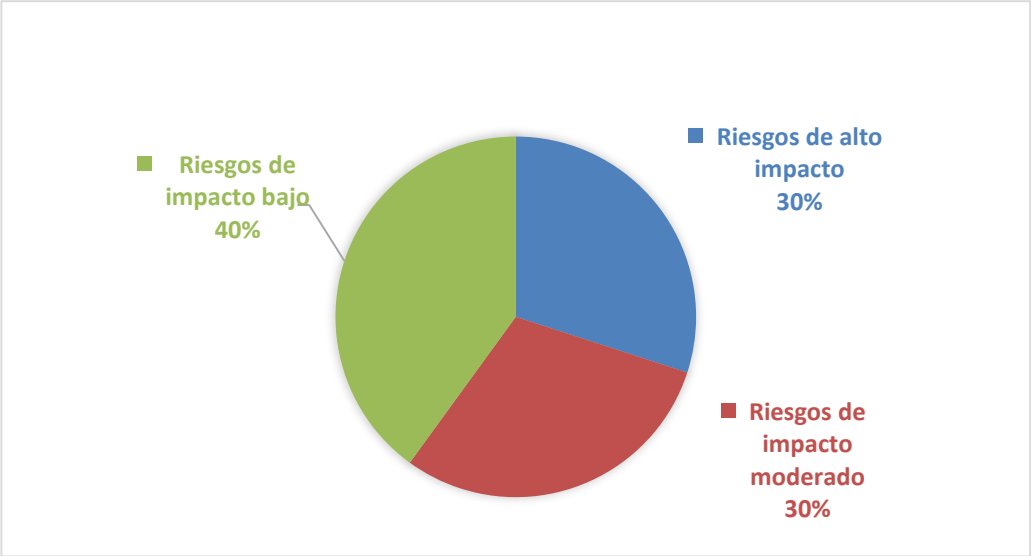
2.3.3.2. Riesgos de impacto moderado

Estos riesgos pueden causar daños considerables, pero no catastróficos al proceso de CI/CD y al sistema. Pueden resultar en la exposición limitada de datos sensibles, interrupciones temporales del servicio, la pérdida o corrupción limitada de datos, o la compro misión parcial de la integridad y la confidencialidad del sistema.

2.3.3.3. Riesgos de impacto bajo

Estos riesgos generalmente causan daños mínimos y son fácilmente manejables. Pueden resultar en la exposición mínima de datos sensibles, interrupciones menores del servicio, la pérdida o corrupción insignificante de datos, o la compro misión superficial de la integridad y la confidencialidad del sistema.

Ilustración 3 Grafica riegos según su impacto



Nota: Fuente Propia

2.4. Análisis y discusión de resultados

Dado que el resultado de nuestra investigación concluye con el análisis e identificación de brechas de seguridad en un entorno en transición hacia DevOps, así como la elaboración de fichas técnicas que detallan el análisis de las configuraciones, descripciones, impacto, alcance y una propuesta de mejora en seguridad para la mitigación de riesgos, llegamos a las siguientes conclusiones.

2.5. Conclusiones y Recomendaciones

2.5.1. Conclusiones

En conclusión, los riesgos de seguridad inherentes a los procesos de Integración Continua/Entrega Continua (CI/CD) representan una preocupación crítica para las empresas de software que dependen de estos procesos para optimizar y acelerar el desarrollo de software. La identificación de las configuraciones inseguras más comunes, en colaboración con las directrices de OWASP, revela un problema creciente en el contexto de la adopción de prácticas DevOps.

Gracias a este análisis, pudimos clasificar estas vulnerabilidades según su naturaleza, impacto y fase del ciclo CI/CD a la que afectan. Este enfoque nos permitió proponer medidas concretas para mitigar el impacto de estas vulnerabilidades y reducir su incidencia en los entornos de desarrollo de software.

2.5.2. Recomendaciones

2.5.2.1. Resumen de las recomendaciones de seguridad para entornos de CI/CD:

2.5.2.2. Controlar el flujo de código

- Implementar reglas para proteger las ramas críticas.
- Limitar el uso de reglas de combinación automática.
- Restringir la ejecución de pipelines de producción.

2.5.2.3. Gestionar identidades y accesos

- Analizar y mapear las identidades en todos los sistemas.
- Eliminar permisos innecesarios.
- Deshabilitar/eliminar cuentas obsoletas.
- Evitar cuentas de usuario locales.

2.5.2.4. Evitar el abuso de la cadena de dependencias

- No permitir que los clientes extraigan paquetes de fuentes no confiables.
- Utilizar un proxy interno para extraer paquetes.
- Verificar la suma de comprobación y las firmas de los paquetes.
- Bloquear las versiones de los paquetes necesarios.

- Implementar controles específicos del marco para bloquear versiones.
- Asegurar que los paquetes privados estén registrados en el ámbito de la organización.
- Evitar publicar nombres de proyectos internos en repositorios públicos.
- Reforzar la seguridad de los sistemas relevantes.

2.5.2.5. Mitigar la ejecución de pipelines envenenadas (PPE)

- Ejecutar pipelines sin revisión en nodos aislados.
- Evaluar la necesidad de desencadenar pipelines en repositorios públicos.
- Implementar reglas de protección de ramas para pipelines confidenciales.
- Revisar los archivos de configuración de CI antes de su ejecución.
- Administrar el archivo de configuración de CI en una rama remota protegida.
- Eliminar permisos innecesarios en el repositorio de SCM.
- Otorgar a cada pipeline solo las credenciales que necesita.

2.5.2.6. Aplicar controles de acceso basados en pipelines (PBAC)

- No utilizar un nodo compartido para pipelines con diferentes niveles de sensibilidad.
- Otorgar a cada pipeline y paso solo los secretos que necesita.
- Revertir el nodo de ejecución a su estado prístino después de cada ejecución de pipeline.
- Limitar los permisos del sistema operativo del usuario que ejecuta el trabajo de pipeline.
- Asegurar que el nodo de ejecución esté parcheado.

- Segmentar la red para permitir que el nodo de ejecución acceda solo a los recursos que necesita.
- Impedir que los scripts de instalación ejecuten comandos con privilegios elevados.

2.5.2.7. Mantener una buena higiene de las credenciales

- Mapear las credenciales en todos los sistemas.
- Evitar compartir credenciales.
- Utilizar credenciales temporales en lugar de estáticas.
- Limitar el uso de credenciales a condiciones predeterminadas.
- Detectar y eliminar secretos de los repositorios de código.
- Otorgar a cada pipeline y paso solo los secretos que necesita.
- Evitar que los secretos se impriman en las salidas de la consola.
- Eliminar secretos de cualquier tipo de artefacto.

2.5.2.8. Configurar los sistemas de forma segura

- Mantener un inventario de sistemas y versiones.
- Restringir el acceso a la red de acuerdo con el principio de acceso mínimo.
- Revisar periódicamente las configuraciones del sistema.
- Otorgar permisos a los nodos de ejecución de pipeline según el principio de privilegios mínimos.

2.5.2.9. Controlar el uso de servicios de terceros

- Establecer procedimientos de aprobación para terceros.
- Monitorizar los permisos de terceros y eliminar los no utilizados.

- Revisar periódicamente los terceros integrados y eliminar los que ya no se usen.

2.5.2.10. Validar la integridad de los artefactos

- Implementar procesos para validar la integridad de los recursos.
- Utilizar firmas de código y software de verificación de artefactos.
- Detectar desviaciones de configuración.
- Validar la integridad de los recursos de terceros.

2.5.2.11. Mejorar el registro y la visibilidad

- Crear un inventario de todos los sistemas en uso.
- Habilitar las fuentes de registro adecuadas.
- Enviar registros a una ubicación centralizada.
- Crear alertas para detectar anomalías y actividades maliciosas

Bibliografía

Bécares, B. (2021). El ataque a SolarWinds, explicado: por qué un ataque a esta empresa desconocida trae de cabeza a grandes corporaciones y gobiernos del mundo. *xataka*. Obtenido de https://www.xataka.com/pro/ataque-a-solarwinds-explicado-que-ataque-a-esta-empresa-desconocida-trae-cabeza-a-grandes-corporaciones-gobiernos-mundo

Fitzgerald, B., & Stol, K. J. (2019). *Continuous Deployment of Microservices: Architecting for DevOps with Kubernetes, Docker, and Azure.*

Fowler, M. (2010). *Continuous Integration. ThoughtWorks.*

GitLab. (2022). Explicación de CI/CD. *GitLab*. Obtenido de https://about.gitlab.com/topics/ci-cd/#ci-cd-explained

HALL, T. (2019). ¿En qué consiste la cultura de DevOps? *ATLASSIAN*. Obtenido de https://www.atlassian.com/es/devops/what-is-devops/devops-culture#:~:text=En%20esencia%2C%20la%20cultura%20de%20DevOps%20implica%20una,hacia%20un%20enfoque%20en%20el%20cliente%20m%C3%A1s%20unificado.

Humble, J. &. (2010). *Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Addison-Wesley.*

Kim, G., & Humble J., &. D. (2018). *The DevOps Handbook: How to Create World-Class Agility, Reliability, & Security in Technology Organizations.*

Mell, P. &. (2011). *The NIST Definition of Cloud Computing. National Institute of Standards and Technology.*

Myers, G. J. (The Art of Software Testing. John Wiley & Sons.). *2011.*

Ocean, D. (2020). *Introducción a prácticas recomendadas de CI/CD.* Digital Ocean. Obtenido de https://www.digitalocean.com/community/tutorials/an-introduction-to-ci-cd-best-practices-es

OWASP. (2022). *Los 10 principales riesgos de seguridad de CI/CD de OWASP.* Obtenido de https://owasp.org/www-project-top-10-ci-cd-security-risks/

Swartout, M. (2020). *Automate Your Software Development Process and Improve Productivity. Packt Publishing.*

ANEXOS

3. Ficha Técnica de las configuraciones más comunes con la identificación de sus brechas de seguridad.

3.1.1. Mecanismos de control de flujo insuficientes

Tabla 1 Mecanismos de control de flujo

Aspecto	Detalle
Título	Mecanismos de control de flujo insuficientes
Definición	Los mecanismos de control de flujo insuficientes se refieren a la capacidad de un atacante que ha obtenido permisos para un sistema dentro del proceso de CI/CD (SCM, CI, repositorio de artefactos, etc.) para enviar por sí solo código malicioso o artefactos por la canalización, debido a la falta de mecanismos que impongan una aprobación o revisión adicional.
Descripción	Los flujos de CI/CD están diseñados para la velocidad, pero esta misma velocidad puede resultar en vulnerabilidades si no se implementan controles adecuados. La falta de mecanismos de control de flujo suficientes puede permitir que un atacante envíe código malicioso o artefactos a través de la canalización sin la revisión adecuada.
Impacto	Un atacante con acceso a sistemas dentro del proceso de CI/CD puede abusar de los mecanismos de control de flujo insuficientes para implementar artefactos maliciosos, lo que podría resultar en la ejecución de código malicioso en producción o en otras partes críticas del sistema.

3.1.2. Gestión inadecuada de identidades y accesos

Tabla 2 Gestión inadecuada de identidades y accesos

Aspecto	Detalle
Título	CICD-SEC-2: Gestión inadecuada de identidades y accesos
Definición	Los riesgos de una gestión inadecuada de identidades y accesos se derivan de las dificultades para gestionar la gran cantidad de identidades repartidas por los diferentes sistemas del ecosistema de ingeniería, desde el control de código fuente hasta la implementación. La existencia de identidades mal administradas, tanto humanas como programáticas, aumenta el potencial y el alcance del daño de su compromiso.
Descripción	Los procesos de entrega de software consisten en múltiples sistemas conectados entre sí con el objetivo de mover el código y los artefactos del desarrollo a la producción. Cada sistema proporciona múltiples métodos de acceso e integración (nombre de usuario y contraseña, token de acceso personal, aplicación de mercado, aplicaciones OAuth, complementos, claves SSH). La complejidad en la administración de las diferentes identidades a lo largo de todo el ciclo de vida de la identidad y en la garantía de que sus permisos estén alineados con el principio de privilegios mínimos puede generar desafíos.
Impacto	La existencia de cientos (o a veces miles) de identidades, tanto humanas como programáticas, en todo el ecosistema de CI/CD, junto con la falta de prácticas sólidas de gestión de identidades y accesos y el uso común de cuentas excesivamente permisivas, conduce a un estado en el que comprometer casi cualquier cuenta de usuario en cualquier sistema podría otorgar capacidades poderosas al entorno y podría servir como transición hacia el entorno de producción.

3.1.3. Abuso de la cadena de dependencias

Tabla 3 Abuso de la cadena de dependencias

Aspecto	Detalle
Título	CICD-SEC-3: Abuso de la cadena de dependencias
Definición	Los riesgos de abuso de la cadena de dependencias se refieren a la capacidad de un atacante para abusar de los errores relacionados con la forma en que las estaciones de trabajo de ingeniería y los entornos de compilación obtienen dependencias de código. El abuso de la cadena de dependencias da como resultado que un paquete malintencionado se recupere y ejecute inadvertidamente localmente cuando se extrae.
Descripción	La gestión de dependencias y paquetes externos utilizados por el código escrito por uno mismo es cada vez más compleja dado el número total de sistemas implicados en el proceso en todos los contextos de desarrollo de una organización. Los paquetes a menudo se obtienen utilizando un cliente dedicado por lenguaje de programación, generalmente desde una combinación de repositorios de paquetes autoadministrados y repositorios SaaS específicos del lenguaje. Los principales vectores de ataque incluyen la confusión de dependencias, el secuestro de dependencias, el typosquatting y el brandjacking.
Impacto	El objetivo de los adversarios que suben paquetes a repositorios públicos de paquetes utilizando una de las técnicas mencionadas anteriormente es ejecutar código malicioso en un host que extrae el paquete. Puede ser la estación de trabajo de un desarrollador o un servidor de compilación que extrae el paquete. Una vez que el código malicioso se está ejecutando, se puede aprovechar para el robo de credenciales y el movimiento lateral dentro del entorno en el que se ejecuta.

3.1.4. Ejecución de Tuberías Envenenadas

Tabla 4 Ejecución de Tuberías Envenenadas

Aspecto	Detalle
Título	CICD-SEC-4: Ejecución de Tuberías Envenenadas (PPE)
Definición	Los riesgos de ejecución de canalización envenenada (PPE) se refieren a la capacidad de un atacante con acceso a los sistemas de control de código fuente, y sin acceso al entorno de compilación, para manipular el proceso de compilación mediante la inyección de código/comandos maliciosos en la configuración de la canalización de compilación, esencialmente "envenenando" la canalización y ejecutando código malicioso como parte del proceso de compilación.
Descripción	El vector PPE abusa de los permisos en un repositorio de SCM, de una manera que hace que una canalización de CI ejecute comandos malintencionados. Los usuarios que tienen permisos para manipular los archivos de configuración de CI u otros archivos en los que se basa el trabajo de canalización de CI pueden modificarlos para que contengan comandos malintencionados y, en última instancia, "envenenar" la canalización de CI que ejecuta estos comandos. Los principales tipos de ejecución de canalización envenenada (PPE) incluyen: - EPP directo (D-PPE): El atacante modifica el archivo de configuración de CI en un repositorio al que tiene acceso, ya sea insertando el cambio directamente en una rama remota desprotegida en el repositorio o enviando una solicitud de incorporación de cambios con el cambio desde una rama o una bifurcación. - EPP indirecto (I-PPE): El atacante inyecta código malintencionado en los archivos a los que hace referencia el archivo de configuración de la canalización, por ejemplo, en scripts, pruebas de código o herramientas automáticas, en lugar de en el archivo de definición de

	canalización directamente. - EPI Público (3PE): En algunos casos, el envenenamiento de las canalizaciones de CI está disponible para atacantes anónimos en Internet, como en repositorios públicos. proyectos de código abierto, que permiten que cualquier usuario contribuya, creando solicitudes de extracción.
Impacto	En un ataque de EPP exitoso, los atacantes ejecutan código malintencionado no revisado en el CI. Esto proporciona al atacante las mismas capacidades y el mismo nivel de acceso que el trabajo de compilación, entre los que se incluyen: - Acceso a cualquier secreto disponible para el trabajo de CI, como secretos insertados como variables de entorno o secretos adicionales almacenados en el CI. - Acceso a activos externos para los que el nodo de trabajo tiene permisos, como archivos almacenados en el sistema de archivos del nodo o credenciales para un entorno en la nube accesible a través del host subyacente. - Capacidad para enviar código y artefactos más adelante en la canalización, bajo la apariencia de código legítimo creado por el proceso de compilación. - Capacidad para acceder a hosts y activos adicionales en la red/entorno del nodo de trabajo.

3.1.5. PBAC (controles de acceso basados en tuberías) insuficientes

Tabla 5 PBAC (controles de acceso basados en tuberías) insuficientes

Aspecto	Detalle
Título	CICD-SEC-5: PBAC (controles de acceso basados en tuberías) insuficientes
Definición	Los nodos de ejecución de canalización tienen acceso a numerosos recursos y sistemas dentro y fuera del entorno de ejecución. Al ejecutar código malicioso dentro de una canalización, los adversarios aprovechan los riesgos insuficientes de PBAC (controles de acceso basados en canalización) para abusar del permiso otorgado a la canalización para moverse lateralmente dentro o fuera del sistema de CI/CD.
Descripción	PBAC incluye controles relacionados con numerosos elementos que tienen que ver con el entorno de ejecución de la canalización, como acceso dentro del entorno de ejecución de la canalización (al código, los secretos, las variables de entorno y otras canalizaciones), permisos para el host subyacente y otros nodos de canalización, y filtros de entrada y salida a Internet. El alcance del daño potencial de un escenario en el que un adversario puede poner en peligro los nodos de ejecución de la canalización o inyectar código malintencionado en el proceso de compilación viene determinado por la granularidad del PBAC en el entorno.
Impacto	Un fragmento de código malintencionado que puede ejecutarse en el contexto del nodo de ejecución de la canalización tiene todos los permisos de la fase de canalización en la que se ejecuta. Puede acceder a secretos, acceder al host subyacente y conectarse a cualquiera de los sistemas a los que tiene acceso la canalización en cuestión. Esto puede dar lugar a la exposición de datos confidenciales, a un movimiento lateral dentro del entorno de CI, con el posible acceso

	a servidores y sistemas fuera del entorno de CI, y a la implementación de artefactos maliciosos en el futuro, incluso en producción.

3.1.6. Higiene insuficiente de las credenciales

Tabla 6 Higiene insuficiente de las credenciales

Aspecto	Detalle
Título	CICD-SEC-6: Higiene insuficiente de las credenciales
Definición	Los riesgos de higiene de credenciales insuficientes se relacionan con la capacidad de un atacante para obtener y usar varios secretos y tokens repartidos por toda la canalización debido a fallas relacionadas con los controles de acceso en torno a las credenciales, la administración insegura de secretos y las credenciales demasiado permisivas.
Descripción	Los entornos de CI/CD están formados por varios sistemas que se comunican y autentican entre sí, lo que crea grandes desafíos en torno a la protección de las credenciales debido a la gran variedad de contextos en los que pueden existir las credenciales. Las credenciales son el objeto más buscado por los adversarios, que buscan usarlas para acceder a recursos de alto valor o para implementar código y artefactos maliciosos.
Impacto	El gran potencial de error humano, junto con las brechas de conocimiento en torno a la gestión segura de credenciales y la preocupación de romper los procesos debido a la rotación de credenciales, ponen los recursos de alto valor de muchas

	organizaciones en riesgo de compromiso debido a la exposición de sus credenciales.

3.1.7. Configuración insegura del sistema

Tabla 7 Configuración insegura del sistema

Aspecto	Detalle
Título	CICD-SEC-7: Configuración insegura del sistema
Definición	Los riesgos de configuración insegura del sistema se derivan de fallas en la configuración de seguridad, la configuración y el endurecimiento de los diferentes sistemas a lo largo de la canalización, lo que a menudo resulta en "frutos al alcance de la mano" para los atacantes que buscan expandir su posición en el entorno.
Descripción	Los entornos de CI/CD se componen de varios sistemas, proporcionados por una variedad de proveedores. Para optimizar la seguridad de CI/CD, se requiere que los defensores pongan un fuerte énfasis tanto en el código y los artefactos que fluyen a través de la canalización, como en la postura y la resistencia de cada sistema individual.
Impacto	Un adversario puede aprovechar una falla de seguridad en uno de los sistemas de CI/CD para obtener acceso no autorizado al sistema o, lo que es peor, comprometer el sistema y acceder al sistema operativo subyacente. Un atacante puede abusar de estas fallas para manipular flujos legítimos de CI/CD, obtener tokens confidenciales y potencialmente acceder a entornos de producción. En algunos escenarios, estas fallas pueden permitir que un atacante se mueva

	lateralmente dentro del entorno y fuera del contexto de los sistemas de CI/CD.

3.1.8. Uso no controlado de servicios de terceros

Tabla 8 Uso no controlado de servicios de terceros

Aspecto	Detalle
Título	CICD-SEC-8: Uso no gobernado de servicios de terceros
Definición	La superficie de ataque de CI/CD consiste en los activos orgánicos de una organización, como el SCM o el CI, y los servicios de terceros a los que se les concede acceso a esos activos orgánicos. Los riesgos que tienen que ver con el uso no controlado de los servicios de terceros se basan en la extrema facilidad con la que se puede conceder acceso a los recursos de un servicio de terceros en los sistemas de CI/CD, lo que amplía eficazmente la superficie de ataque de la organización.
Descripción	Es raro encontrar una organización que no tenga numerosos terceros conectados a sus sistemas y procesos de CI/CD. Su facilidad de implementación, combinada con su valor inmediato, ha hecho de los terceros una parte integral del día a día de la ingeniería. Los métodos de incorporación o concesión de acceso a terceros son cada vez más diversos y las complejidades asociadas a su implementación están disminuyendo.
Impacto	La falta de gobernanza y visibilidad en torno a las implementaciones de terceros impide que las organizaciones mantengan RBAC dentro de sus sistemas de CI/CD. Dado lo permisivos que tienden a ser los

	terceros, las organizaciones son tan seguras como los terceros que implementan.

3.1.9. Validación incorrecta de la integridad del artefacto

Tabla 9 Validación incorrecta de la integridad del artefacto

Aspecto	Detalle
Título	CICD-SEC-9: Validación incorrecta de la integridad del artefacto
Definición	Los riesgos de validación de integridad de artefactos incorrectos permiten a un atacante con acceso a uno de los sistemas en el proceso de CI/CD insertar código o artefactos maliciosos (aunque aparentemente benignos) en la canalización, debido a mecanismos insuficientes para garantizar la validación del código y los artefactos.
Descripción	Los procesos de CI/CD constan de varios pasos, responsables en última instancia de llevar el código desde la estación de trabajo de un ingeniero hasta la producción. Hay múltiples recursos que se introducen en cada paso, combinando recursos y artefactos internos con paquetes y artefactos de terceros obtenidos de ubicaciones remotas. El hecho de que el recurso final dependa de múltiples fuentes distribuidas en los diferentes pasos, proporcionadas por múltiples contribuyentes, crea múltiples puntos de entrada a través de los cuales se puede manipular este recurso definitivo. Si un recurso manipulado fue capaz de infiltrarse con éxito en el proceso de entrega, sin levantar ninguna sospecha ni encontrar ninguna puerta de seguridad, lo más probable es que continúe fluyendo a través de la tubería, hasta la producción, bajo la apariencia de un recurso legítimo.

Impacto	Un adversario con un punto de apoyo dentro del proceso de entrega de software puede abusar de la validación incorrecta de la integridad de los artefactos para enviar un artefacto malicioso a través de la canalización, lo que en última instancia resulta en la ejecución de código malicioso, ya sea en sistemas dentro del proceso de CI/CD o peor, en producción.

3.1.10. Registro y visibilidad insuficientes

Tabla 10 Registro y visibilidad insuficientes

Aspecto	Detalle
Título	CICD-SEC-10: Registro y visibilidad insuficientes
Definición	Los riesgos de registro y visibilidad insuficientes permiten que un adversario lleve a cabo actividades maliciosas dentro del entorno de CI/CD sin ser detectado durante ninguna fase de la cadena de eliminación del ataque, incluida la identificación de las TTP (técnicas, tácticas y procedimientos) del atacante como parte de cualquier investigación posterior al incidente.

Descripción	La existencia de sólidas capacidades de registro y visibilidad es esencial para la capacidad de una organización para prepararse, detectar e investigar un incidente relacionado con la seguridad. Si bien las estaciones de trabajo, los servidores, los dispositivos de red y las aplicaciones empresariales y de TI clave suelen cubrirse en profundidad dentro de los programas de registro y visibilidad de una organización, a menudo no es el caso de los sistemas y procesos en entornos de ingeniería. Dada la cantidad de vectores de ataque potenciales que aprovechan los entornos y procesos de ingeniería, es imperativo que los equipos de seguridad desarrollen las capacidades adecuadas para detectar estos ataques tan pronto como ocurran. Dado que muchos de estos vectores implican aprovechar el acceso programático frente a los diferentes sistemas, un aspecto clave para hacer frente a este reto es crear niveles sólidos de visibilidad en torno al acceso humano y programático. Dada la naturaleza sofisticada de los vectores de ataque de CI/CD, existe el mismo nivel de importancia tanto para los registros de auditoría de los sistemas, por ejemplo, el acceso de usuarios, la creación de usuarios, la modificación de permisos, como para los registros de aplicación, por ejemplo, el evento push a un repositorio, la ejecución de compilaciones, la carga de artefactos.
Impacto	Con los adversarios cambiando gradualmente su enfoque a los entornos de ingeniería como un medio para lograr sus objetivos, las organizaciones que no garantizan los controles adecuados de registro y visibilidad en torno a esos entornos, pueden no detectar una infracción y enfrentar grandes dificultades en la mitigación/corrección debido a las capacidades mínimas de investigación. El tiempo y los datos son los bienes más valiosos para una organización que está siendo atacada. La existencia de todas las fuentes de datos relevantes en una ubicación centralizada puede ser la diferencia entre un

	resultado exitoso y devastador en un escenario de respuesta a incidentes.