

OWASP Top 10 CI/CD

Seguridad y Calidad en
Aplicaciones Web / Calidad en
Aplicaciones Móviles

Basado en material
OWASP Top 10 CI/CD Security Risks



Universidad Nacional
de La Matanza

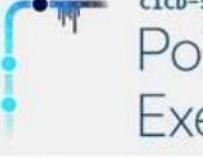
Profesores

- **Profesores a cargo de los cursos:**

Mgter. Lic. Walter Ureta

Mgter. Ing. Pablo Pomar

Seguridad en CI/CD (DevOps/DevSecOps)

Top 10 CI/CD Security Risk	
	CICD-SEC-1 Insufficient Flow Control Mechanisms
	CICD-SEC-2 Inadequate Identity and Access Management
	CICD-SEC-3 Dependency Chain Abuse
	CICD-SEC-4 Poisoned Pipeline Execution (PPE)
	CICD-SEC-5 Insufficient PBAC (Pipeline-Based Access Controls)
	CICD-SEC-6 Insufficient Credential Hygiene
	CICD-SEC-7 Insecure System Configuration
	CICD-SEC-8 Ungoverned usage of 3 rd party services
	CICD-SEC-9 Improper Artifact Integrity Validation
	CICD-SEC-10 Insufficient logging and visibility



1 – 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 debido a la falta de mecanismos que hagan cumplir una aprobación o revisión adicional**.

Descripción

Se puede **crear un nuevo código en la máquina de un desarrollador y pasar a la producción en cuestión de minutos**, a menudo **con plena confianza en la automatización y una participación humana mínima**. Al ver que los procesos de CI/CD son esencialmente la vía hacia los entornos de producción altamente cerrados y protegidos, **las organizaciones introducen continuamente medidas y controles destinados a garantizar que ninguna entidad (humana o aplicación) pueda enviar código o artefactos a través de la canalización sin tener que someterse a un conjunto estricto de revisiones y aprobaciones**.

1 – Mecanismos de Control de Flujo Insuficientes

Impactos

Un atacante con acceso al SCM, CI o sistemas que se encuentran en el pipeline **puede abusar de los insuficientes mecanismos de control de flujo para implementar artefactos maliciosos. Una vez creados, los artefactos se envían a través del pipeline, potencialmente hasta la producción, sin ninguna aprobación o revisión.**

Por ejemplo, un atacante puede:

- Abusar de una regla de fusión automática en el CI que fusiona automáticamente las solicitudes de incorporación de cambios que cumplen con un conjunto predefinido de requisitos, lo que genera código malicioso no revisado.
- Abusar de la falta de reglas de protección de las ramas (branchs), por ejemplo, excluir usuarios o ramas específicas para eludir la protección de ramas y enviar código malicioso no revisado.
- Acceder a la producción y cambiar directamente el código o la infraestructura de la aplicación (por ejemplo, la función AWS Lambda), sin ninguna aprobación/verificación adicional.

1 – Mecanismos de Control de Flujo Insuficientes

Recomendaciones

Establecer mecanismos de control de flujo en el pipeline para garantizar que ninguna entidad individual (humana/programática) pueda enviar código y artefactos confidenciales a través del pipeline sin verificación o validación externa.

Esto se puede lograr implementando las siguientes medidas (entre otras):

- Limite el uso de reglas de fusión automática y asegúrese de que, dondequiera que estén en uso, sean aplicables a la cantidad mínima de contextos. Revise minuciosamente el código de todas las reglas de combinación automática para asegurarse de que no se puedan omitir y evite importar código de terceros en el proceso de combinación automática.
- Cuando corresponda, evite que las cuentas desencadenen canalizaciones de desarrollo e implementación de producción sin aprobación o revisión adicional.
- Preferir permitir que los artefactos fluyan a través de la canalización solo con la condición de que hayan sido creados por una cuenta de servicio de CI preaprobada. Evite que los artefactos que han sido cargados por otras cuentas fluyan a través de la canalización sin una revisión y aprobación secundarias.

2 – Gestión inadecuada de identidad y acceso

Definición

Los riesgos de **una gestión de acceso e identidad inadecuada** se derivan de las **dificultades para gestionar la gran cantidad de identidades repartidas en los diferentes sistemas del ecosistema** de ingeniería, desde el control de origen hasta la implementación. **La existencia de identidades mal administradas, tanto cuentas humanas como programáticas, aumenta el potencial y el alcance del daño del compromiso.**





2 – Gestión inadecuada de identidad y acceso

Descripción

Los procesos de entrega de software consisten en múltiples sistemas conectados entre sí con el objetivo de mover el código y los artefactos desde el desarrollo hasta la producción. Cada sistema proporciona múltiples métodos de acceso e integración (nombre de usuario y contraseña, token de acceso personal, aplicación de mercado, aplicaciones de autenticación, complementos, claves SSH). Los diferentes tipos de cuentas y métodos de acceso pueden tener potencialmente su propio método de aprovisionamiento único, conjunto de políticas de seguridad y modelo de autorización. Esta complejidad crea desafíos en la gestión de las diferentes identidades a lo largo de todo el ciclo de vida de la identidad y garantiza que sus permisos estén alineados con el principio de privilegio mínimo.

Estas identidades son utilizadas principalmente por ingenieros que requieren la flexibilidad para poder crear cambios importantes en el código y la infraestructura.

Algunas de las principales preocupaciones y desafíos en torno a la gestión de identidades y accesos dentro del ecosistema de CI/CD incluyen (entre otros):

- Identidades excesivamente permisivas: Se debe garantizar que a cada identidad humana y de aplicación se le hayan otorgado solo los permisos necesarios y solo en los repositorios reales a los que necesita acceder.
- Identidades obsoletas: empleados/sistemas que no están activos y/o ya no requieren acceso pero no ha sido desaprovisionados su cuenta humana y programática contra todos los sistemas de CI/CD.





2 – Gestión inadecuada de identidad y acceso

Impactos

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 sólidas prácticas de administración de acceso e identidad y el uso común de cuentas demasiado permisivas, conduce a un estado en el que comprometer casi cualquier cuenta de usuario en cualquier sistema, podría otorgar poderosas capacidades en el entorno y podría servir como transición al entorno de producción.

Recomendaciones (entre otras)

- Llevar a cabo un análisis y un mapeo continuos de todas las identidades en todos los sistemas dentro del ecosistema de ingeniería. Para cada identidad, asigne el proveedor de identidad, el nivel de permisos otorgados y el nivel de permisos realmente utilizados. Asegúrese de que todos los métodos de acceso programático estén cubiertos en el análisis.
- Eliminar los permisos que no sean necesarios para el trabajo en curso de cada identidad en los diferentes sistemas del entorno.
- Determinar un período aceptable para deshabilitar/eliminar cuentas obsoletas y deshabilitar/eliminar cualquier identidad que haya superado el período predeterminado de inactividad.



3 – Abuso sobre la Cadena de Dependencias

Definición

Los riesgos de abuso de la cadena de dependencia se refieren a la **capacidad de un atacante para abusar de las fallas relacionadas con la forma en que las estaciones de trabajo de ingeniería y los entornos de construcción obtienen las dependencias del código**. El abuso de la cadena de dependencia da como resultado que un paquete malicioso se obtenga y ejecute localmente sin darse cuenta cuando se le realice un pull.



3 – Abuso sobre la Cadena de Dependencias

Descripción

La gestión de dependencias y paquetes externos utilizados por el código escrito por uno mismo se está volviendo cada vez más compleja dada la cantidad total de sistemas involucrados en el proceso en todos los contextos de desarrollo de una organización.

Muchas organizaciones hacen todo lo posible para **detectar el uso de paquetes con vulnerabilidades conocidas y realizar análisis estáticos de código escrito por ellos mismos y de terceros**.

Los principales vectores de ataque en este contexto son:

- **Dependency confusion** (Confusión de dependencia): publicación de paquetes maliciosos en repositorios públicos con el mismo nombre que los nombres de los paquetes internos, en un intento de engañar a los clientes para que descarguen el paquete malicioso en lugar del privado.
- **Dependency Hijacking** (Secuestro de dependencias): obtener el control de la cuenta de un mantenedor de paquetes en el repositorio público para cargar una nueva versión maliciosa de un paquete ampliamente utilizado, con la intención de comprometer a los clientes desprevenidos que extraen la última versión del paquete.
- **Typosquatting** (similitud de tipo/nombre): publicación de paquetes maliciosos con nombres similares a los de los paquetes populares con la esperanza de que un desarrollador escriba mal el nombre de un paquete y obtenga involuntariamente el paquete con error tipográfico.
- **BrandHijacking** (Robo de marca): publicación de paquetes maliciosos de manera consistente con la convención de nomenclatura u otras características del paquete de una marca específica, en un intento de que los desarrolladores desprevenidos obtengan estos paquetes debido a una asociación falsa con la marca de confianza.



3 – Abuso sobre la Cadena de Dependencias

Impactos

El objetivo de los adversarios que **cargan paquetes en repositorios de paquetes públicos utilizando una de las técnicas antes mencionadas** es ejecutar código malicioso en un **host que extrae el paquete**.

Esto podría ser la estación de trabajo de un desarrollador o un servidor de compilación que extrae el paquete. **Una vez que se ejecuta el código malicioso**, se puede aprovechar para el robo de credenciales y el movimiento lateral dentro del entorno en el que se ejecuta.

Otro escenario potencial es que el código malicioso del atacante llegue a los entornos de producción desde el servidor de compilación. En muchos casos, el paquete malicioso continuaría manteniendo la funcionalidad original y segura que esperaba el usuario, lo que resultaría en una menor probabilidad de detección.

3 – Abuso sobre la Cadena de Dependencias Recomendaciones (cont.)

Existe una amplia gama de métodos de mitigación que son específicos para la configuración de los diferentes clientes específicos del lenguaje y la forma en que se utilizan los proxies internos y los repositorios de paquetes externos.

Dicho esto, todos los controles recomendados comparten los mismos principios rectores:

- **No se debe permitir que ningún cliente extraiga paquetes de código para obtener paquetes directamente de Internet o de fuentes no confiables.** En su lugar, se deben implementar los siguientes controles:
 - Siempre que los paquetes de terceros se extraigan de un repositorio externo, asegúrese de que **todos los paquetes se extraigan a través de un proxy interno en lugar de hacerlo directamente desde Internet**. Esto permite implementar controles de seguridad adicionales en la capa de proxy, así como proporcionar capacidades de investigación en torno a los paquetes extraídos, en caso de un incidente de seguridad.
 - Cuando corresponda, **prohibir la extracción de paquetes directamente desde repositorios externos**. Configure todos los clientes para extraer paquetes de repositorios internos, que contengan paquetes previamente examinados, y establezca un mecanismo para verificar y hacer cumplir esta configuración de cliente.
- Habilite la **verificación de la suma de control y la verificación de la firma para los paquetes extraídos**.



3 – Abuso sobre la Cadena de Dependencias

Recomendaciones (cont.)

- Evite configurar clientes para extraer la última versión de un paquete. Es preferible configurar una versión previamente examinada o rangos de versiones. Utilice las técnicas específicas del marco para "bloquear" continuamente la versión del paquete requerida en su organización a una versión estable y segura.
- Alcances:
 - Asegúrese de que todos los paquetes privados estén registrados bajo el alcance de la organización.
 - Asegúrese de que todo el código que haga referencia a un paquete privado use el alcance del paquete.
 - Asegúrese de que los clientes se vean obligados a obtener paquetes que están bajo el alcance de su organización únicamente desde su registro interno.
- Cuando los scripts de instalación se ejecutan como parte de la instalación del paquete, asegúrese de que exista un contexto separado para esos scripts, que no tenga acceso a secretos y otros recursos confidenciales disponibles en otras etapas del proceso de compilación.
- Asegúrese de que los proyectos internos siempre contengan archivos de configuración de administradores de paquetes (por ejemplo, .npmrc en NPM) dentro del repositorio de código del proyecto, para anular cualquier configuración insegura que pueda existir en un cliente que busca el paquete.



3 – Abuso sobre la Cadena de Dependencias

Recomendaciones (cont.)

- **Evitar publicar nombres de proyectos internos en repositorios públicos.**
- Como regla general, dada la cantidad de administradores de paquetes y configuraciones en uso simultáneamente, **la prevención completa del abuso de la cadena por parte de terceros está lejos de ser trivial.**
- Por lo tanto, se recomienda **asegurarse de que se pone un nivel adecuado de atención en torno a la detección, el seguimiento y la mitigación para garantizar que, en caso de un incidente, se identifique lo más rápido posible y tenga la mínima cantidad de daño potencial.**
- En este contexto, **todos los sistemas relevantes deben fortalecerse adecuadamente de acuerdo con las pautas bajo el riesgo "CICD-SEC-7: Configuración Insegura del sistema".**

4 – Ejecución de un Pipeline Envenenado

Definición

Los riesgos de ejecución de pipeline 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 construcción, **para manipular el proceso de construcción mediante la inyección de códigos/comandos maliciosos en la configuración de la pipeline de construcción**, esencialmente "envenenamiento" del pipeline y la ejecución de código malicioso como parte del proceso de compilación..



4 – Ejecución de un Pipeline Envenenado

Descripción

El vector PPE abusa de los permisos contra un repositorio SCM, de una manera que hace que una pipeline de CI ejecute comandos maliciosos.

Los usuarios que tienen permisos para manipular los archivos de configuración de CI, u otros archivos en los que se basa el trabajo de pipeline de CI, pueden modificarlos para que contengan comandos maliciosos y, en última instancia, "envenenar" la pipeline de CI que ejecuta estos comandos.

Los pipelines que ejecutan código no revisado, por ejemplo, aquellos que se activan directamente a partir de solicitudes de incorporación de cambios o confirmaciones en ramas arbitrarias del repositorio, son más susceptibles a PPE.

Una vez que puede ejecutar código malicioso dentro de la tubería de CI, el atacante puede realizar una amplia gama de operaciones maliciosas, todo dentro del contexto de la identidad del pipeline.



4 – Ejecución de un Pipeline Envenenado

Descripción (cont.)

- **PPE directo (D-PPE)** En un escenario D-PPE, el atacante modifica el archivo de configuración de CI en un repositorio al que tiene acceso
- **PPE indirecto (I-PPE)** En ciertos casos, la posibilidad de D-PPE no está disponible para un adversario con acceso a un repositorio SCM.
- **PPE público (3PE)** la ejecución de un ataque PPE requiere acceso al repositorio que aloja el archivo de configuración del pipeline o a los archivos a los que hace referencia. En la mayoría de los casos, el permiso para hacerlo se otorgaría a los miembros de la organización, principalmente ingenieros. Por lo tanto, los atacantes normalmente tendrían que estar en posesión del permiso de un ingeniero en el repositorio para ejecutar un ataque PPE directo o indirecto.

Si el pipeline de CI de un repositorio público ejecuta código no revisado sugerido por usuarios anónimos, es susceptible a un ataque PPE público o, en resumen, 3PE. Esto también expone activos internos, como secretos de proyectos privados, en los casos en que la pipeline del repositorio público vulnerable se ejecuta en la misma instancia de CI que los privados.



4 – Ejecución de un Pipeline Envenenado

Impactos

En un ataque PPE exitoso, los atacantes ejecutan código malicioso no revisado en el CI. Esto proporciona al atacante las mismas habilidades y el mismo nivel de acceso que el trabajo de compilación, que incluye:

- **Acceso a cualquier secreto disponible para el trabajo de CI**, como secretos injectados como variables de entorno o secretos adicionales almacenados en el CI. Al ser responsables de crear código e implementar artefactos, los sistemas de CI/CD generalmente contienen docenas de credenciales y tokens de alto valor, como para un proveedor de nube, registros de artefactos y el propio SCM.
- **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 de nube accesible a través del host subyacente**.
- **Capacidad de enviar código y artefactos** más adelante en el pipeline, **bajo la apariencia de código legítimo generado por el proceso de compilación**.
- **Capacidad para acceder a hosts y activos adicionales en la red/entorno del nodo de trabajo**.

4 – Ejecución de un Pipeline Envenenado

Recomendaciones

Prevenir y mitigar el vector de ataque PPE implica múltiples medidas que abarcan los sistemas SCM y CI:

- **Asegurar que los pipelines que ejecutan código no revisado se ejecuten en nodos aislados, no expuestos a secretos ni entornos confidenciales.**
- **Evaluar la necesidad de desencadenar pipelines en repositorios públicos de colaboradores externos.** Siempre que sea posible, absténgase de ejecutar pipelines que se originen en bifurcaciones y considere agregar controles, como requerir aprobación manual para la ejecución del pipeline.
- **Para evitar la manipulación del archivo de configuración de CI para ejecutar código malintencionado en el pipeline, se debe revisar cada archivo de configuración de CI antes de ejecutar el pipeline.**
- **Eliminar los permisos otorgados en el repositorio de SCM a los usuarios que no los necesitan.**
- **Cada pipeline sólo debe tener acceso a las credenciales que necesita para cumplir su propósito. Las credenciales deben tener los privilegios mínimos requeridos.**

5 – Insuficientes Controles de Acceso basados en Pipeline

Definición

Los nodos de ejecución de un pipeline tienen acceso a numerosos recursos y sistemas dentro y fuera del entorno de ejecución. **Al ejecutar código malicioso dentro de un pipeline, los atacantes aprovechan los riesgos de PBAC (Controles de acceso basados en pipeline) insuficientes para abusar del permiso otorgado a la pipeline para moverse lateralmente dentro o fuera del sistema CI/CD.**

5 – Insuficientes Controles de Acceso basados en Pipeline Descripción

Los pipeline son el "motor" de CI/CD. Los **nodos** que ejecutan pipeline llevan a cabo los comandos especificados en la configuración del pipeline y, al hacerlo, **realizan una amplia gama de actividades confidenciales**:

- **Acceder al código fuente, compilarlo y probarlo.**
- **Obtener secretos de varias ubicaciones, como variables de entorno, bóvedas, servicios de identidad basados en la nube dedicados (como el servicio de metadatos de AWS) y otras ubicaciones.**
- **Crear, modificar y desplegar artefactos.**

Dada la naturaleza altamente sensible y crítica de cada pipeline, es **imperativo limitar cada pipeline al conjunto exacto de datos y recursos a los que necesita acceder**.

PBAC incluye controles relacionados con numerosos elementos que tienen que ver con el entorno de ejecución del pipeline:

- **Acceso dentro del entorno de ejecución del pipeline: a código, secretos, variables de entorno y otras canalizaciones.**
- **Permisos para el host subyacente y otros nodos del pipeline.**
- **Filtros de ingreso y egreso a internet.**



5 – Insuficientes Controles de Acceso basados en Pipeline

Impactos

Una pieza de código malicioso que puede ejecutarse en el contexto del nodo de ejecución del pipeline tiene todos los permisos de la etapa del pipeline 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 el pipeline en cuestión.**

Esto puede dar lugar a la **exposición de datos confidenciales, movimiento lateral dentro del entorno de CI, accediendo potencialmente a servidores y sistemas fuera del entorno de CI, y la implementación de artefactos maliciosos en el proceso, incluida la producción.**

El alcance del daño potencial de un escenario en el que un atacante puede comprometer los nodos de ejecución del pipeline o injectar código malicioso en el proceso de compilación **está determinado por la granularidad del PBAC en el entorno.**





5 – Insuficientes Controles de Acceso basados en Pipeline

Recomendaciones

- No utilice un nodo compartido para pipelines con diferentes niveles de sensibilidad/que requieran acceso a diferentes recursos.
- Asegurar que los secretos que se utilizan en los sistemas de CI/CD tengan un alcance que permita que cada pipeline y tenga acceso sólo a los secretos que necesita.
- Revertir el nodo de ejecución a su estado original después de cada ejecución de pipeline.
- Asegurar que al usuario del sistema operativo que ejecuta el trabajo de pipeline se le hayan otorgado permisos de sistema operativo en el nodo de ejecución de acuerdo con el principio de privilegio mínimo.
- Asegurar que el nodo de ejecución tenga los parches correspondientes.
- Asegurar que la segmentación de la red en el entorno en el que se ejecuta el trabajo esté configurada para permitir que el nodo de ejecución acceda solo a los recursos que requiere dentro de la red. Siempre que sea posible, abstenerse de otorgar acceso ilimitado a Internet para construir nodos.



6 – Insuficiente “Higiene” de Credenciales

Definición

Los riesgos de no realizar insuficiente "higiene" de credenciales se relacionan con la **capacidad de un atacante para obtener y usar varios secretos y tokens distribuidos a lo largo del pipeline debido a fallas que tienen que ver con los controles de acceso alrededor de las credenciales, la gestión insegura y demasiado permisiva de los secretos y las credenciales.**





6 – Insuficiente “Higiene” de Credenciales

Descripción

Los entornos de CI/CD están construidos con múltiples sistemas que se comunican y se 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.

La aplicación utiliza las **credenciales de la aplicación en tiempo de ejecución**, los pipelines utilizan las credenciales de los sistemas de producción **para implementar la infraestructura, los artefactos y las aplicaciones en la producción**, los ingenieros usan las credenciales **como parte de sus entornos de prueba y dentro de su código y artefactos**.

Esta variedad de contextos, junto con la gran cantidad de métodos y técnicas para **almacenarlos y usarlos, crea un gran potencial para el uso inseguro de las credenciales**.

Algunas fallas importantes que afectan la "higiene" de las credenciales:

- Código que contiene credenciales que se envían a una de las ramas (branch) de un SCM
- Credenciales utilizadas de forma insegura dentro del repositorio de procesos de compilación e implementación
- Credenciales en capas de imágenes de contenedores
- Credenciales impresas en la salida de la consola
- Credenciales no rotadas



6 – Insuficiente “Higiene” de Credenciales

Impactos

Las credenciales son el objeto más buscado por los atacantes, que buscan usarlas **para acceder a recursos de alto valor o para implementar códigos y artefactos maliciosos**. En este contexto, los entornos de ingeniería brindan a los atacantes múltiples vías para obtener credenciales. **El gran potencial de error humano, junto con las lagunas de conocimiento sobre la gestión segura de credenciales y la preocupación de interrumpir los procesos debido a la rotación de credenciales, ponen en riesgo los recursos de alto valor de muchas organizaciones debido a la exposición de sus credenciales.**

6 – Insuficiente “Higiene” de Credenciales

Recomendaciones

- Establecer **procedimientos para mapear continuamente las credenciales** que se encuentran en los diferentes sistemas del ecosistema de ingeniería, desde el código hasta la implementación. **Asegurar que cada conjunto de credenciales siga el principio de privilegio mínimo y que se le haya otorgado el conjunto exacto de permisos que necesita el servicio que lo usa.**
- Evitar compartir el mismo conjunto de credenciales en varios contextos. Esto aumenta la complejidad de lograr el principio de privilegio mínimo.
- Optar por usar credenciales temporales sobre credenciales estáticas. En caso de que sea necesario usar credenciales estáticas, establecer un procedimiento para rotar periódicamente todas las credenciales estáticas y detectar las obsoletas.
- Asegurar que los secretos que se utilizan en los sistemas de CI/CD tengan un alcance que permita que cada pipeline y paso tenga acceso solo a los secretos que necesita.
- Usar opciones de proveedores integradas o herramientas de terceros para evitar que los secretos se impriman en las salidas de la consola de compilaciones futuras.
Asegurar de que todas las salidas existentes no contengan secretos.

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 hardening de los diferentes sistemas a lo largo del pipeline (por ejemplo, SCM, CI, repositorio de artefactos). Situación que favorece a los atacantes porque se les facilita el trabajo.



7 – Configuración Insegura del Sistema

Descripción

Los entornos de CI/CD se componen de múltiples sistemas, proporcionados por una variedad de proveedores. **Para optimizar la seguridad de CI/CD, los defensores deben poner mucho énfasis tanto en el código y los artefactos que fluyen a través del pipeline como en la postura y la resistencia de cada sistema individual.**

Ejemplos de **posibles defectos de hardening**:

- Un sistema y/o componente autogestionado que utiliza una **versión desactualizada o carece de parches de seguridad importantes**.
- Un sistema que tiene **controles de acceso a la red demasiado permisivos**.
- Un sistema con **configuraciones de sistema inseguras**. Las configuraciones suelen determinar las características de seguridad clave que tienen que ver con la autorización, los controles de acceso, el registro y más. En muchos casos, el **conjunto predeterminado de configuraciones no es seguro y requiere optimización**.
- Un sistema con una **higiene de credenciales inadecuada**; por ejemplo, credenciales predeterminadas que no están deshabilitados, tokens programáticos demasiado permisivos.



7 – Configuración Insegura del Sistema

Impactos

Un atacante puede aprovechar una falla de seguridad en uno de los sistemas 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 de CI/CD legítimos, 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.



7 – Configuración Insegura del Sistema

Recomendaciones

- Mantener un **inventario de sistemas y versiones en uso**, incluido el mapeo de un **propietario designado para cada sistema**. Verificar continuamente las **vulnerabilidades conocidas** en estos componentes. Si hay un parche de seguridad disponible, **actualizar el componente vulnerable**. De lo contrario, considere **eliminar el componente/sistema o reduzca el impacto potencial** de explotar la vulnerabilidad **restringiendo el acceso al sistema o la capacidad del sistema para realizar operaciones confidenciales**.
- Garantizar que el **acceso de red** a los sistemas esté **alineado con el principio de acceso mínimo**.
- Establezca un proceso para **revisar periódicamente todas las configuraciones del sistema** en busca de cualquier configuración que pueda tener un efecto en la postura de seguridad del sistema y asegúrese de que todas las configuraciones sean óptimas.



8 – Uso no controlados de Servicios de terceros

Definición

La superficie de ataque de CI/CD consiste en los servicios de terceros a los que se les otorga acceso a esos activos orgánicos como SCM o CI. Los riesgos que tienen que ver con el uso no controlado de servicios de terceros se basan en la extrema facilidad con la que se puede otorgar acceso a un servicio de terceros a los recursos en los sistemas de CI/CD, expandiendo efectivamente 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 convertido a los terceros en una parte integral del día a día de la ingeniería. Los métodos para incorporar u otorgar acceso a terceros son cada vez más diversos y las complejidades asociadas con su implementación están disminuyendo. Por ejemplo, las aplicaciones de terceros se pueden conectar a través de uno o más de estos 5 métodos:

- Aplicación GitHub
- Aplicación OAuth
- Aprovisionamiento de un token de acceso ...
- Aprovisionamiento de una clave SSH ...
- Configuración de eventos de webhook para enviar a terceros.

8 – Uso no controlados de Servicios de terceros

Impactos

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.

La implementación insuficiente de RBAC y los privilegios mínimos en torno a terceros, junto con un gobierno y diligencia mínimos en torno al proceso de implementaciones de terceros, crean un aumento significativo de la superficie de ataque de la organización.

Recomendaciones

Los controles de gobernanza en torno a los servicios de terceros deben implementarse en cada etapa del ciclo de vida de uso de terceros:

- **Aprobación:** establezca procedimientos de investigación para garantizar que los terceros a los que se otorga acceso a los recursos en cualquier parte del ecosistema de ingeniería sean aprobados antes de que se les otorgue acceso al entorno, y que el nivel de permiso que se les otorgue esté alineado con el principio de privilegio mínimo.

8 – Uso no controlados de Servicios de terceros

Recomendaciones (cont)

- **Integración:** introduzca controles y procedimientos para mantener una visibilidad continua sobre todos los terceros integrados en los sistemas de CI/CD, incluidos:
 - Método de integración. Asegúrese de que se cubran todos los métodos de integración para cada sistema (incluidas las aplicaciones de mercado, complementos, aplicaciones OAuth, tokens de acceso programático, etc.).
 - Nivel de permiso otorgado a la tercera parte.
 - Nivel de permiso actualmente en uso por el tercero.
- **Visibilidad sobre el uso continuo:** asegúrese de que cada tercero esté limitado y limitado a los recursos específicos a los que requiere acceso y elimine los permisos no utilizados o redundantes. Los terceros que se integran como parte del proceso de compilación deben ejecutarse dentro de un contexto de ámbito con acceso limitado a los secretos y el código, y con filtros de entrada y salida estrictos.
- **Desaprovisionamiento:** revise periódicamente todos los terceros integrados y elimine los que ya no estén en uso.

9 – Validación Inapropiada de Integridad de Artefactos

Definición

Los riesgos de validación inadecuada de integridad de artefactos permiten que un atacante con acceso a uno de los sistemas en el proceso de CI/CD envíe códigos o artefactos maliciosos (aunque aparentemente benignos) por la canalización, debido a mecanismos insuficientes para garantizar la validación del código y los artefactos.

Descripción

Los procesos de CI/CD consisten en múltiples 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 alimentan 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 repartidas en los diferentes pasos, proporcionadas por múltiples contribuyentes, crea múltiples puntos de entrada a través de los cuales se puede manipular este recurso final.

Si un recurso manipulado pudo infiltrarse con éxito en el proceso de entrega, sin despertar sospechas ni encontrar puertas de seguridad, lo más probable es que continúe fluyendo a través de la tubería, hasta la producción, bajo la apariencia de un recurso legítimo.



9 – Validación Inapropiada de Integridad de Artefactos

Impactos

Un atacante, con un punto de apoyo en el proceso de entrega de software puede abusar de la validación de integridad de artefactos incorrecta 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.

Recomendaciones

La prevención de riesgos de validación inadecuada de integridad de artefactos requiere una colección de medidas, a través de diferentes sistemas y etapas dentro de la cadena de entrega de software. Considere los siguientes controles:

- Implementar procesos y tecnologías para validar la integridad de los recursos desde el desarrollo hasta la producción. Cuando se genera un recurso, el proceso incluirá la firma de ese recurso mediante una infraestructura de firma de recursos externa. Antes de consumir el recurso en los pasos posteriores de la canalización, la integridad del recurso debe validarse con la autoridad de firma.

9 – Validación Inapropiada de Integridad de Artefactos Recomendaciones (cont)

Algunas medidas predominantes a considerar en este contexto:

- Firma de código: las soluciones de SCM brindan la capacidad de firmar compromisos utilizando una clave única para cada colaborador. Esta medida se puede aprovechar para evitar que las confirmaciones sin firmar fluyan por la canalización.
- Software de verificación de artefactos: el uso de herramientas para firmar y verificar el código y los artefactos proporciona una manera de evitar que se entregue software no verificado en la canalización.
- Detección de cambios de configuración: medidas destinadas a detectar cambios de configuración.
- Recursos de terceros obtenidos de canalizaciones de compilación/implementación debe seguir una lógica similar: antes al uso de recursos de terceros, se debe calcular el hash del recurso y referencia cruzada contra el hash oficial publicado del proveedor de recursos.

10 – Insuficiente logueo y visibilidad

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 destrucción 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 que una organización pueda 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 comerciales y de TI clave generalmente se tratan en profundidad dentro de los programas de registro y visibilidad de una organización, a menudo no es el caso con los sistemas y procesos en entornos de ingeniería.

Dada la cantidad de posibles vectores de ataque 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.



10 – Insuficiente logueo y visibilidad

Impactos

Fallar en detectar una brecha y enfrentar grandes dificultades en la mitigación / remediación debido a la mínima capacidades investigativas.

El tiempo y los datos son los bienes más valiosos para una organización bajo ataque.

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.

Recomendaciones

Hay varios elementos para lograr suficiente registro y visibilidad:

- Mapeo del entorno: Identifique y cree un inventario de todos los sistemas en uso dentro de la organización, que contenga cada instancia de estos sistemas.
- Identificar y habilitar las fuentes de registro adecuadas.
- Envío de registros a una ubicación centralizada (p. ej., SIEM), para respaldar la agregación y correlación de registros entre diferentes sistemas para detección e investigación.
- Crear alertas para detectar anomalías y posibles actividades maliciosas, tanto en cada sistema por sí mismo como anomalías en el proceso de envío de código, que involucra múltiples sistemas y requiere un conocimiento más profundo en los procesos internos de construcción e implementación.



¿Dudas?

DIIT



Departamento de Ingeniería e
Investigaciones Tecnológicas



Universidad Nacional
de La Matanza

Muchas gracias



DIIT

Departamento de Ingeniería e
Investigaciones Tecnológicas



Universidad Nacional
de La Matanza