Google Cloud

Informe técnico
Enero de 2022

Entrega software de forma segura

Índice

Capítulo

01 Información sobre la cadena de suministro de software

Capítulo

02 Estándares y frameworks para toda la industria

Índice

Capítulo

03 Servicios administrados para cada etapa

Capítulo

04 Primeros pasos

Capítulo

05 Recursos útiles

Chapter

01 Información sobre la cadena de suministro de software

Panorama de seguridad actual

La velocidad y el tiempo de salida al mercado son las prioridades principales para las organizaciones en todo el mundo que crean software y aplicaciones con el objetivo de satisfacer las necesidades de sus clientes. Estos imperativos estratégicos son la fuerza que impulsó el enorme crecimiento de los contenedores como plataforma preferida. Durante el último año, muchas de estas organizaciones cosecharon los beneficios de los contenedores, como la aceleración de la salida al mercado, una mayor disponibilidad y seguridad, una mejor escalabilidad y menos costos, por lo que comenzaron a considerar la adopción de un enfoque sin servidores.

Si bien las soluciones de software reducen el tiempo que demora entregar una funcionalidad nueva o incluso un producto nuevo, muchas de las prácticas de seguridad actuales no pueden seguirle el ritmo al aumento de velocidad, lo que genera uno de los siguientes tres problemas:

Los procesos existentes ralentizan a los desarrolladores, lo que provoca retrasos

Los equipos de seguridad y operaciones generan riesgos que exponen a las organizaciones a amenazas

Los equipos de desarrollo trabajan con procesos de seguridad existentes para cumplir con los plazos, lo que los hace vulnerables

En los últimos años, se perpetró una gran cantidad de violaciones de seguridad clasificadas como ataques a la “cadena de suministro de software”.

Log4Shell fue una vulnerabilidad peligrosa del software Apache Log4j que se identificó en diciembre de 2021. Con una puntuación máxima de CVSS de 10, esta vulnerabilidad fue devastadora, en particular debido a la popularidad de Log4j, un framework de registro basado en Java. Dos factores contribuyeron a su gravedad: en primer lugar, era muy fácil de explotar y permitía ejecutar por completo el código de forma remota; y, en segundo lugar, a menudo se encontraba a muchas capas de profundidad en la estructura de dependencias, por lo que era sencillo pasarla por alto.

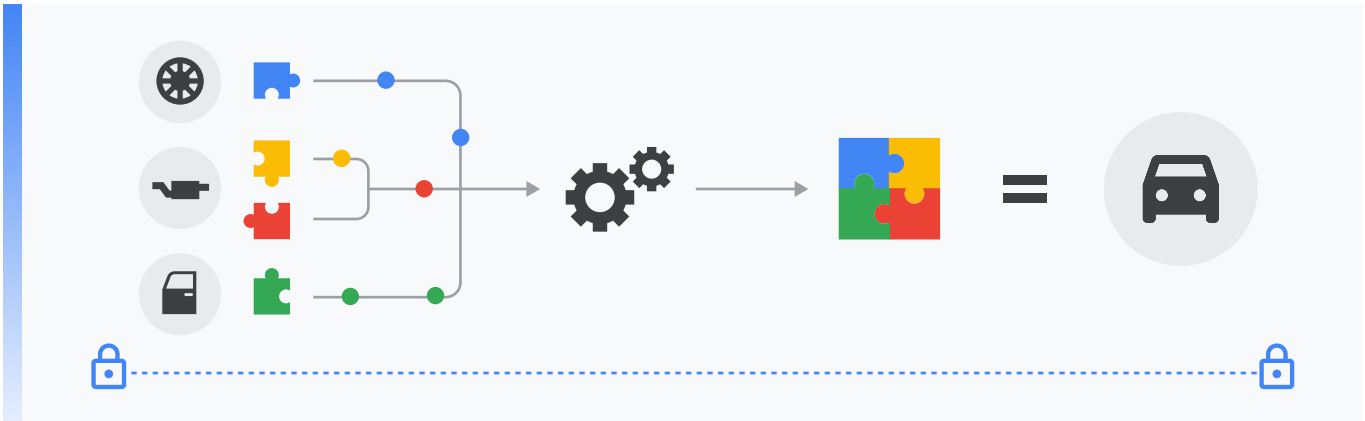
Solarwinds, una empresa desarrolladora de software de administración de TI, sufrió el ataque de hackers de un estado-nación que insertaron código malicioso en las compilaciones oficiales de software de código abierto que se utilizaba en la empresa. Esta actualización maliciosa se envió a 18,000 clientes, incluidos el Tesoro y el Departamento de Comercio de los EE.UU.

Kaseya, otro proveedor de software de administración de TI, sufrió un ataque por medio de una vulnerabilidad de día cero que puso en riesgo el servidor de Kaseya VSA y envió un script malicioso para distribuir un ransomware que encriptó todos los archivos en los sistemas afectados.

La necesidad urgente de responder a estos y otros incidentes similares hicieron que la Casa Blanca emitiera un decreto en mayo de 2021, donde se exige que las organizaciones que operan con el gobierno federal mantengan ciertos estándares de seguridad de software.

La cadena de suministro de software

En muchos sentidos, el término “cadena de suministro de software” es muy adecuado: los procesos que se transitan para crear una cadena de suministro de software son muy similares a los que se usan en la fabricación de un automóvil.



Un fabricante de automóviles obtiene varias piezas estándares, fabrica sus propios componentes y, luego, los une mediante un proceso altamente automatizado. Para garantizar la seguridad de sus operaciones, el fabricante se asegura de que cada componente externo provenga de una fuente confiable. Los componentes propios se prueban de manera exhaustiva para garantizar que no existan problemas de seguridad. Por último, el ensamblaje se realiza a través de un proceso confiable que da como resultado el automóvil terminado.

La cadena de suministro de software es similar en muchos aspectos. Un fabricante de software obtiene componentes de terceros, que suelen ser de código abierto por naturaleza, para que ejecuten funciones específicas, y también desarrolla su propio software, que es su propiedad intelectual principal. Luego, el código se somete a un proceso de compilación que combina estos componentes en artefactos implementables, que luego se implementan en producción.

El eslabón débil de la cadena

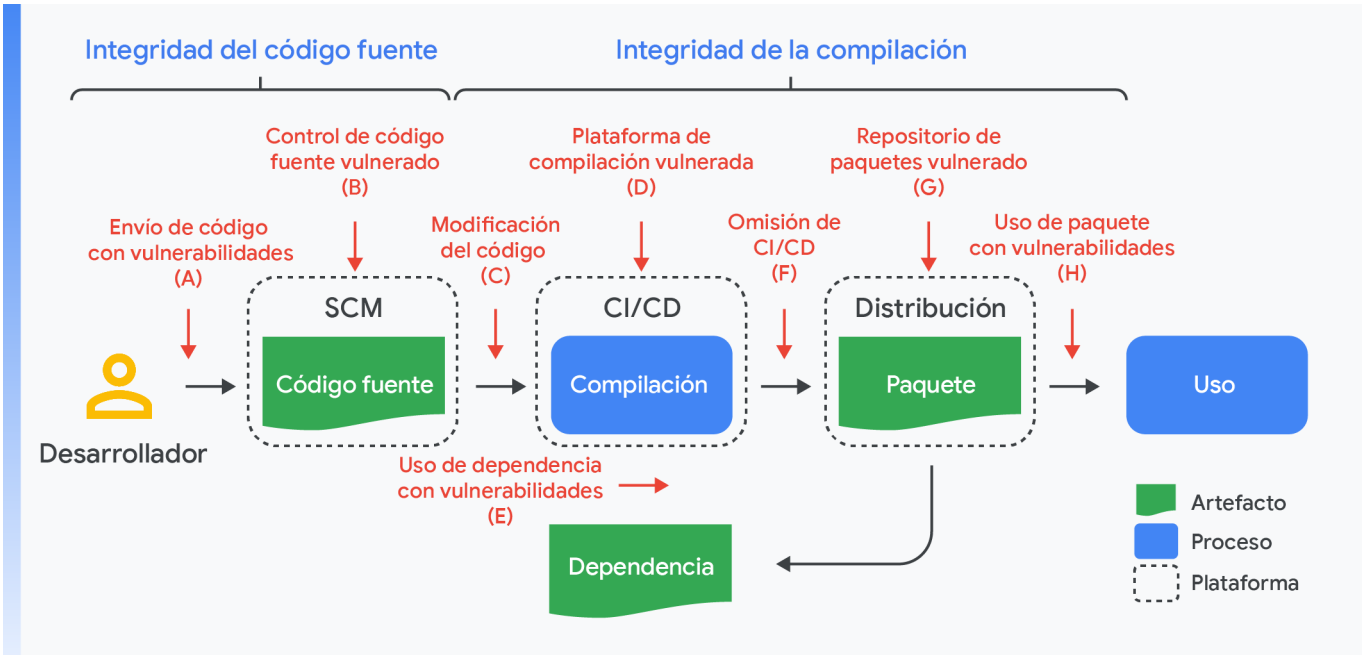
Solo se necesita un enlace no seguro para vulnerar la cadena de suministro de software

Al igual que en los ataques de alto perfil del año pasado, cada uno de los pasos del proceso puede generar una debilidad que puede ser aprovechada por los atacantes.

Por ejemplo, el paquete de npm promedio tiene 12 dependencias directas y aproximadamente 300 dependencias indirectas. Además, sabemos que casi el 40% de los paquetes de npm publicados dependen de códigos con vulnerabilidades conocidas.

Es posible que esas vulnerabilidades no sean responsables de que el código sea inseguro; por ejemplo, la vulnerabilidad podría formar parte de una biblioteca que nunca se usa. Sin embargo, se deben verificar.

La magnitud de este problema es monumental. Incluso si no se le aplicara un parche a solo una de estas vulnerabilidades, se generaría una oportunidad para que los hackers ingresen a tu cadena de suministro de software.



Estos son algunos ejemplos de ataques que aprovecharon las vulnerabilidades en cada una de las etapas que se muestran en el diagrama anterior.

	Amenaza	Ejemplo conocido
A	Envío de un código con vulnerabilidades al repositorio de código fuente	Parches infectados de Linux: un investigador intentó introducir vulnerabilidades de manera intencional en el kernel de Linux mediante parches en la lista de distribución.
B	Plataforma de control de código fuente vulnerada	PHP: el atacante vulneró el servidor git autoalojado de PHP e insertó dos commits maliciosos.
C	Compilación con proceso oficial, pero desde un código que no coincide con el control de código fuente	Webmin: el atacante modificó la infraestructura de la compilación para que use archivos de origen que no coinciden con el control de código fuente.
D	Plataforma de compilación vulnerada	SolarWinds: el atacante vulneró la plataforma de compilación e instaló un implante para insertar comportamientos maliciosos durante cada compilación.
E	Uso de dependencia con vulnerabilidades (es decir, A-H, de forma recurrente)	event-stream: el atacante agregó una dependencia inocua y luego la actualizó para incluir un comportamiento malicioso. La actualización no coincidió con el código enviado a GitHub (es decir, ataque F).
F	Carga de un artefacto no compilado por el sistema de CI/CD	Codecov: el atacante usó credenciales filtradas para subir un artefacto malicioso a un depósito de GCS, desde el cual los usuarios descargan directamente.
G	Repositorios de paquetes vulnerados	Ataques a la duplicación de paquetes: el investigador ejecutó duplicaciones para varios repositorios populares de paquetes, que se podrían haber usado para entregar paquetes maliciosos.
H	Engaño al consumidor para que use un paquete con vulnerabilidades	Errores ortográficos de Browserify: el atacante cargó un paquete malicioso con un nombre similar al original.

Una cadena más sólida: el liderazgo de Google Cloud en código abierto

En Google, llevamos décadas creando aplicaciones a escala global. Con el tiempo, configuramos con código abierto muchos de nuestros proyectos internos para ayudar a aumentar la velocidad de los desarrolladores. Al mismo tiempo, desarrollamos varios procesos internos para ayudar a proteger la experiencia de software.

Estas son algunas de nuestras iniciativas para fortalecer las cadenas de suministro de software en todas partes

Mayor inversión – En agosto de 2020, anunciamos una inversión $10,000 millones en los próximos cinco años para fortalecer la seguridad cibernética, lo que incluye expandir los programas de confianza cero, contribuir a la protección de la cadena de suministro de software y mejorar la seguridad del código abierto.

Niveles de la cadena de suministro para artefactos de software (SLSA) – SLSA es un framework de extremo a extremo diseñado para la integridad de la cadena de suministro. Es un equivalente de código abierto de muchos de los procesos que implementamos internamente en Google. SLSA proporciona una procedencia auditable de lo que se creó y cómo se hizo.

Investigación y evaluación de DevOps (DORA) – Nuestro equipo de DORA realizó un programa de investigación de siete años en el que se validaron diferentes capacidades técnicas, culturales, de procesos y de medición que impulsan la entrega de software y el rendimiento de la organización.

Base de seguridad de código abierto – Cofundamos Open Source Security Foundation en 2019, un foro interdisciplinario sobre la seguridad en la cadena de suministro.

Allstar – Allstar es una aplicación de GitHub instalada en organizaciones o repositorios para definir y aplicar políticas de seguridad. Esto permite la aplicación continua de prácticas recomendadas de seguridad para los proyectos de GitHub.

Cuadros de evaluación de código abierto – Estos paneles usan métricas de evaluación, como políticas de seguridad definidas, procesos de revisión de código y una cobertura de pruebas continua mediante herramientas de fuzzing y de análisis de código estático para proporcionar una puntuación del riesgo en proyectos de código abierto.

Creemos que existen dos elementos necesarios para superar el problema de seguridad en la cadena de suministro de software:

1. Estándares y frameworks para toda la industria.
2. Servicios administrados que implementen estos estándares mediante principios de mínimo privilegio en una arquitectura de confianza cero. Una arquitectura de confianza cero es aquella en la que no se confía de manera inherente en ninguna persona, dispositivo o red: toda confianza se debe ganar, lo que permite acceder a la información.

Analizaremos cada uno de ellos en los siguientes capítulos.

Chapter

02 Estándares y frameworks para toda la industria

A fin de comprender los principios necesarios para proteger la cadena de suministro de software, comencemos con SLSA.

En la actualidad, SLSA hace referencia a un conjunto de lineamientos de seguridad que se pueden implementar de forma incremental y que se estableció por consenso de la industria. En su formato final, SLSA difiere de una mera lista de prácticas recomendadas en lo que respecta a su aplicabilidad: admitirá la creación automática de metadatos auditables que se pueden ingresar en motores de políticas para otorgar una “certificación de SLSA” a un paquete o una plataforma de compilación en particular.

El framework SLSA fue diseñado para ser incremental y útil, con el objetivo de proporcionar beneficios de seguridad en cada etapa. Una vez que un artefacto califica en el nivel más alto, los consumidores pueden estar seguros de que no se alteró y puede rastrearse de forma segura hasta su fuente (algo que es difícil, si no imposible, de hacer con la mayoría del software hoy en día).

SLSA consta de cuatro niveles, de los cuales SLSA 4 representa un estado final ideal. Los niveles inferiores representan hitos incrementales con las garantías de integridad incrementales correspondientes. Actualmente, los requisitos se definen de la siguiente manera:

SLSA 1 requiere que el proceso de compilación esté completamente planificado y automatizado, y que genere una procedencia. La procedencia son los metadatos sobre cómo se compiló un artefacto, incluido el proceso de compilación, el código fuente de nivel superior y las dependencias. Conocer la procedencia permite que los consumidores de software tomen decisiones de seguridad basadas en riesgos. Aunque la procedencia en SLSA 1 no brinda protección contra la manipulación, ofrece un nivel básico de identificación de la fuente del código y puede ayudar en la administración de vulnerabilidades.

SLSA 2 requiere usar el control de versiones y un servicio de compilación alojado que genere una procedencia autenticada. Estos requisitos adicionales le otorgan más confianza al consumidor sobre el origen del software. En este nivel, la procedencia evita la manipulación en la medida en que el servicio de compilación sea de confianza. SLSA 2 también proporciona una ruta de optimización sencilla a SLSA 3.

SLSA 3 también requiere que las plataformas de origen y compilación cumplan con estándares específicos para garantizar la auditabilidad de la fuente y la integridad de la procedencia, respectivamente. SLSA 3 ofrece protecciones mucho más sólidas contra la manipulación que los niveles anteriores, ya que previene clases específicas de amenazas, como la contaminación entre compilaciones.

SLSA 4 es actualmente el nivel más alto, y requiere una revisión de todos los cambios por parte de dos personas, además de un proceso de compilación hermético y reproducible. La opinión de dos personas es una práctica recomendada en la industria para detectar errores y disuadir comportamientos inadecuados. Las compilaciones herméticas garantizan que la lista de dependencias de la procedencia esté completa. Las compilaciones reproducibles, aunque no sean estrictamente obligatorias, proporcionan muchos beneficios de auditabilidad y confiabilidad. En términos generales, SLSA 4 otorga al consumidor un alto nivel de confianza de que el software no se manipuló. Más información sobre estos niveles propuestos en el repositorio de GitHub, incluidos los correspondientes requisitos de origen y compilación/procedencia.

La cadena de suministro de software puede dividirse en cinco fases distintas: código, compilación, paquetes, implementación y ejecución. Abordaremos cada una de estas fases en función de nuestro enfoque de seguridad.

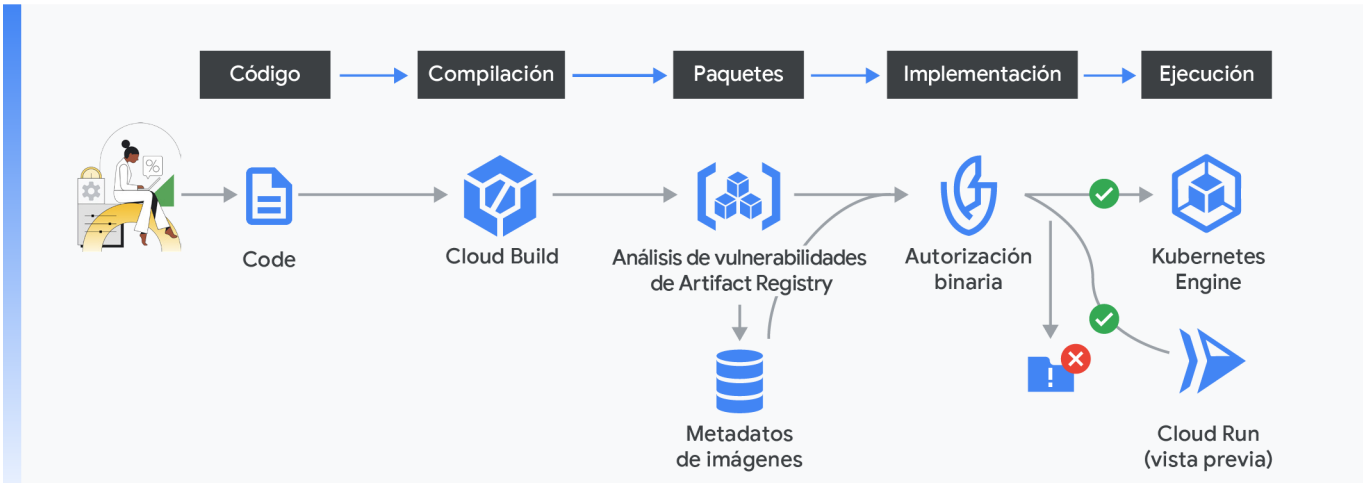
Chapter

03 Servicios administrados para cada etapa

En Google Cloud, proporcionamos herramientas completamente administradas, desde la generación de código y la compilación hasta la implementación y la ejecución, aplicando en forma predeterminada las prácticas recomendadas y los estándares ya mencionados.

Para proteger la cadena de suministro de software, se debe establecer, verificar y mantener una cadena de confianza que establezca la procedencia del código y garantice que lo que se ejecute en producción sea correcto. En Google, esto se logra mediante certificaciones que se generan y verifican a lo largo del proceso de desarrollo e implementación de software, lo que permite un nivel de seguridad ambiental mediante aspectos como la revisión del código, la procedencia verificada del código y la aplicación de políticas. En conjunto, estos procesos nos ayudan a minimizar el riesgo en la cadena de suministro de software y, a su vez, mejorar la productividad de los desarrolladores.

En la base, tenemos servicios comunes de infraestructura segura, como la administración de identidades y accesos y los registros de auditoría. Luego, protegemos la cadena de suministro de software con un método que permite definir, verificar y aplicar certificaciones en todo el ciclo de vida del software.
Analicemos con más detalle cómo lograr la seguridad ambiental en un proceso de desarrollo mediante políticas y procedencias en Google Cloud.
.



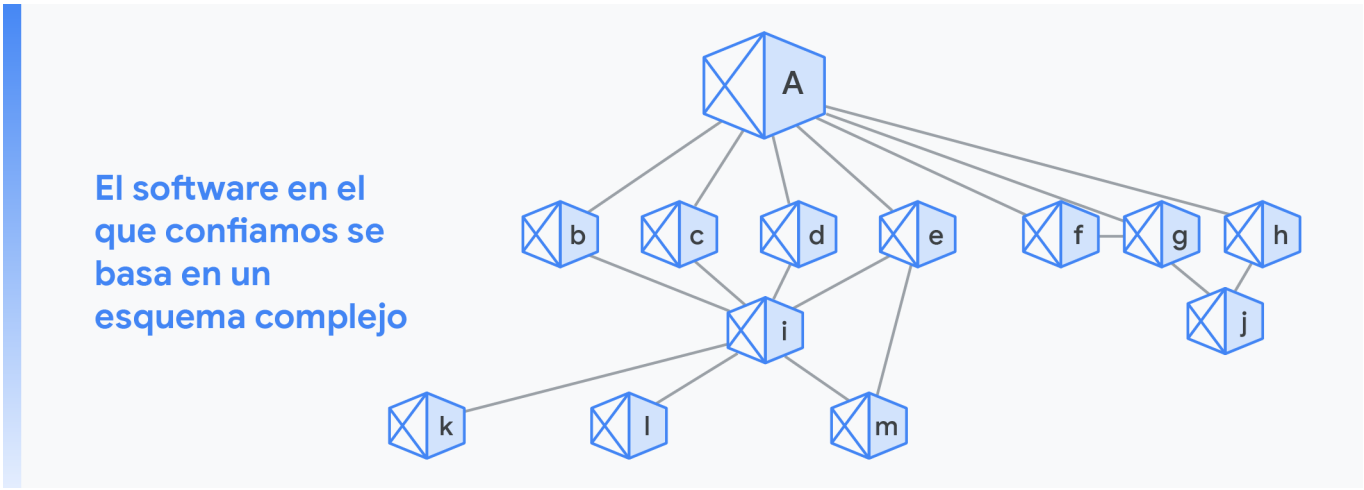
Fase 1: Código

La protección de la cadena de suministro de software empieza cuando los desarrolladores comienzan a diseñar una aplicación y a generar el código. Esto incluye tanto el software propio como los componentes de código abierto, cada uno de los cuales presenta desafíos.

Dependencias y software de código abierto

El código abierto les permite a los desarrolladores compilar proyectos más rápido para que las organizaciones puedan ser más ágiles y productivas. Sin embargo, el software de código abierto no es perfecto y, si bien nuestra industria depende de él, a menudo tenemos muy poca información sobre sus dependencias y los diferentes niveles de riesgo que conlleva. Para la mayoría de las empresas, el riesgo proviene principalmente de vulnerabilidades o licencias.

El software, los paquetes, las imágenes base y otros artefactos de código abierto de los que se depende forman la base de la “cadena de confianza” de una empresa.



Por ejemplo, pensemos en una organización que compila el software “a”. En este diagrama, se muestra la cadena de confianza; es decir, la cantidad de dependencias implícitas del proyecto. En la imagen, de “b” a “h” son dependencias directas, mientras que de “i” a “m” son dependencias indirectas.

Ahora, consideremos que hay una vulnerabilidad en lo más profundo del árbol de dependencias. El problema puede presentarse en muchos componentes muy rápido. Además, las dependencias cambian con bastante frecuencia: en un día promedio, 40,000 paquetes de npm experimentan un cambio en sus dependencias.

Open Source Insights es una herramienta creada por Google Cloud que proporciona un gráfico de dependencias transitivo para visualizar dependencias propias y externas, todo en un árbol de dependencias. Open Source Insights se actualiza continuamente con asesorías de seguridad, información sobre licencias y otros datos de seguridad en varios lenguajes, todo en un solo lugar. Cuando se usa en conjunto con cuadros de evaluación de código abierto, que proporcionan una puntuación de riesgo para los proyectos de código abierto, Open Source Insights ayuda a los desarrolladores a tomar mejores decisiones en función de los millones de paquetes de código abierto disponibles.

Con el fin de abordar este problema, es clave enfocarse en las dependencias como código. A medida que estas dependencias se desplazan hacia el final de la cadena de suministro, es más difícil inspeccionarlas. Para proteger las dependencias, recomendamos comenzar con el suministro:

- Utilizar herramientas como Open Source Insights y los cuadros de evaluación de OSS para comprender mejor las dependencias.
- Analizar y verificar todo el código, los paquetes y las imágenes base con un proceso automatizado que sea una parte clave del flujo de trabajo.
- Controlar cómo las personas acceden a estas dependencias. Es fundamental controlar estrictamente los repositorios de código fuente y de código abierto, con restricciones en torno a la revisión exhaustiva del código y requisitos de auditoría

Más adelante, analizaremos los procesos de compilación e implementación con más detalle, pero también es importante verificar la procedencia de la compilación, utilizar un entorno de compilación seguro y garantizar que las imágenes estén firmadas y que luego se validen en el momento de la implementación.

Los desarrolladores también pueden usar varias prácticas de codificación segura:

- Automatizar pruebas
- Usar lenguajes de software seguros para la memoria
- Ordenar revisiones del código
- Garantizar la autenticidad de los commits
- Identificar códigos maliciosos con anticipación
- Evitar la exposición de información sensible
- Exigir la generación de registros y resultados de la compilación
- Aprovechar la gestión de licencias

Fase 2: Compilación

El siguiente paso para proteger la cadena de suministro de software implica establecer un entorno de compilación seguro a gran escala. En esencia, el proceso de compilación comienza con la importación del código fuente en uno de varios lenguajes desde un repositorio y la ejecución de compilaciones para cumplir con las especificaciones que se establecen en los archivos de configuración

Los proveedores de servicios de nube, como Google, brindan acceso a un entorno de compilación administrado y actualizado que permite compilar imágenes a cualquier escala que se necesite.

A medida que se avanza en el proceso de compilación, se debe tener en cuenta lo siguiente:

- ¿Los secretos están protegidos durante el proceso de compilación y en etapas posteriores?
- ¿Quién tiene acceso a los entornos de compilación?
- ¿Qué ocurre con los vectores de ataque relativamente nuevos o los riesgos de robo de datos?

Para desarrollar un entorno de compilación seguro, es necesario comenzar con los secretos. Son críticos y relativamente fáciles de proteger. En primer lugar, las empresas deben asegurarse de que sus secretos nunca sean de texto plano y, en la medida de lo posible, no formen parte de su compilación. De hecho, deben asegurarse de que estén encriptados y de que las compilaciones estén parametrizadas para consultar secretos externos y usarlos según sea necesario. Esto también simplifica la rotación periódica de los secretos y minimiza el impacto de cualquier filtración.

El siguiente paso es configurar los permisos para la compilación. Existen varios usuarios y cuentas de servicio involucrados en un proceso de compilación. Por ejemplo, es posible que algunos usuarios necesiten gestionar secretos, mientras que otros tengan que administrar el proceso de compilación agregando o modificando etapas. Además, es posible que otros solo necesiten ver los registros.

En este proceso, es importante seguir estas prácticas recomendadas:

- En primer lugar, tenemos el principio de privilegio mínimo. Se deben implementar permisos específicos según los necesiten los usuarios y las cuentas de servicio a fin de realizar sus trabajos con eficacia.
- Luego, es necesario conocer cómo interactúan los usuarios y las cuentas de servicio, y comprender claramente la cadena de responsabilidad, desde la configuración de una compilación hasta su ejecución y sus efectos.

A continuación, a medida que se avanza en este proceso, se deben establecer límites en torno a la compilación en la medida de lo posible y, luego, usar la automatización para escalar verticalmente a través de procesos de configuración como código y parametrización. Esto permite auditar de manera eficaz cualquier cambio en el proceso de compilación. Además, se deben satisfacer las necesidades de cumplimiento a través del acceso de aprobación a implementaciones y compilaciones sensibles, las solicitudes de extracción para los cambios en la infraestructura y las revisiones manuales periódicas de los registros de auditoría.

Por último, la empresa debe asegurarse de que la **red** se adapte a sus necesidades. En la mayoría de los casos, es mejor alojar el propio código fuente en redes privadas con firewalls. Google Cloud brinda acceso a funciones como los grupos privados de Cloud Build, un entorno de compilación sin servidores y bloqueado en el propio perímetro de red privada, y también a funciones como los Controles del servicio de VPC, que permiten evitar el robo de datos de la propiedad intelectual.

Autorización binaria

Si bien IAM es imprescindible y un punto de partida lógico, no es infalible. Las filtraciones de credenciales representan un riesgo de seguridad grave, por lo que, para reducir la dependencia de IAM, es posible cambiar a un sistema basado en certificaciones que sea menos propenso a errores. Google usa un sistema llamado autorización binaria, que permite que solo se implementen cargas de trabajo de confianza.

El servicio de autorización binaria establece, verifica y mantiene una cadena de confianza mediante certificaciones y verificaciones de políticas durante todo el proceso. En esencia, la autorización binaria genera firmas criptográficas (certificaciones) a medida que el código y otros artefactos se desplazan hacia producción y, luego, antes de la implementación, se verifican estas certificaciones en función de las políticas.

Con Google Cloud Build, se captura un conjunto de certificaciones y se agrega a la cadena de confianza general. Por ejemplo, se generan certificaciones para qué tareas ejecutadas, qué herramientas y procesos de compilación usados, etcétera. En particular, Cloud Build ayuda a alcanzar el nivel SLSA 1 mediante la captura de la fuente de la configuración de compilación, que se puede usar para validar que la compilación se haya realizado mediante scripts. Las compilaciones con scripts son más seguras que las manuales, y son necesarias para el nivel SLSA 1. Además, la procedencia de una compilación y otras certificaciones se pueden buscar mediante el resumen de la imagen de contenedor, que crea una firma única para cualquier imagen y que también es obligatorio para el nivel SLSA 1.

Fase 3: Paquetes

Cuando se complete la compilación, tendremos una imagen de contenedor casi lista para la etapa de producción. Es fundamental contar con una ubicación segura donde almacenar las imágenes, ya que eso puede evitar la manipulación de las imágenes existentes y la carga de imágenes no autorizadas. Es probable que el administrador de paquetes necesite imágenes tanto para compilaciones propias como de código abierto, así como para los paquetes de lenguaje que usan las aplicaciones.

Artifact Registry de Google Cloud proporciona un repositorio de este tipo. Artifact Registry permite que la organización administre imágenes de contenedor y paquetes de lenguajes (como Maven y npm) en un solo lugar. Está completamente integrado a las herramientas y los entornos de ejecución de Google Cloud y es compatible con protocolos de artefactos nativos. Esto facilita la integración con las herramientas de CI/CD mientras se configuran las canalizaciones automáticas.

Al igual que en el paso de compilación, es fundamental garantizar que los permisos de acceso a Artifact Registry estén bien definidos y sigan los principios de privilegio mínimo. Además de restringir el acceso no autorizado, el repositorio de paquetes puede proporcionar mucho más valor. Por ejemplo, Artifact Registry incluye el análisis de vulnerabilidades para escanear imágenes y garantizar que sean seguras de implementar. Este servicio analiza imágenes en función de una base de datos de vulnerabilidades, que se actualiza constantemente a fin de evaluar las nuevas amenazas, y puede brindar alertas cuando se encuentra una vulnerabilidad.

En este paso, se generan metadatos adicionales, incluida una certificación para determinar si los resultados de la vulnerabilidad de un artefacto cumplen con ciertos umbrales de seguridad. Luego, esta información se almacena en nuestro servicio de análisis, que estructura y organiza los metadatos del artefacto, lo que facilita su acceso a la autorización binaria. Esta función permite evitar de forma automática que se implementen imágenes riesgosas en Google Kubernetes Engine (GKE).

Fases 4 y 5: Implementación y ejecución

Las dos fases finales de la cadena de suministro de seguridad de software son las de implementación y ejecución. Si bien estos son dos pasos independientes, tiene sentido pensar en ellos como una forma de garantizar que solo las compilaciones autorizadas lleguen a la etapa de producción.

En Google, desarrollamos prácticas recomendadas para determinar qué tipo de compilaciones se deben autorizar. Estas comienzan con la garantía de la integridad de la cadena de suministro, de modo que produzca solo artefactos que sean confiables. Luego, incluye la administración de vulnerabilidades como parte del ciclo de vida de la entrega de software. Por último, unimos esos dos elementos a fin de aplicar flujos de trabajo en función de políticas para asegurar la integridad y el análisis de vulnerabilidades.

Al llegar a esta etapa, significa que las empresas ya han transitado por las fases de código, compilación y paquetes. Las certificaciones capturadas a lo largo de la cadena de suministro se pueden verificar para comprobar su autenticidad mediante la autorización binaria. En el modo de aplicación, una imagen se implementa solo cuando las certificaciones cumplen con las políticas de la organización y, en el modo de auditoría, se registran los incumplimientos de políticas y se activan alertas. También la autorización binaria sirve para evitar que se ejecuten las compilaciones a menos que se hayan compilado mediante el proceso aprobado de Cloud Build. La autorización binaria garantiza que solo se implemente el código correctamente revisado y autorizado.

Implementar imágenes en un entorno de ejecución confiable es esencial. GKE, nuestra plataforma administrada de Kubernetes, adopta un enfoque que prioriza la seguridad de los contenedores.

GKE se encarga de muchas de las inquietudes sobre la seguridad del clúster que necesitan resolver las empresas. Las actualizaciones automáticas de clústeres permiten aplicar parches y actualizar automáticamente

Kubernetes mediante el uso de canales de versiones. El arranque seguro, los nodos protegidos y las verificaciones de integridad garantizan que los componentes del clúster y el kernel del nodo no se hayan modificado y ejecuten lo que la organización desea, y que los nodos maliciosos no puedan ingresar al clúster. Por último, el procesamiento confidencial permite ejecutar clústeres con nodos cuya memoria está encriptada para que se pueda mantener la confidencialidad de los datos mientras se procesan. Además, con la encriptación de datos en reposo y en tránsito a través de la red, GKE proporciona un entorno muy seguro, privado y confidencial para ejecutar cargas de trabajo alojadas en contenedores.

Además, GKE también habilita una mayor seguridad para las aplicaciones, mediante la administración de certificados para los balanceadores de cargas, la identidad de cargas de trabajo y capacidades de red avanzadas con una poderosa manera de configurar y proteger los ingresos en el clúster. GKE ofrece entornos de zona de pruebas para ejecutar aplicaciones no confiables y, al mismo tiempo, proteger el resto de las cargas de trabajo.

Con Autopilot de GKE, las prácticas recomendadas y las funcionalidades de seguridad de GKE se implementan automáticamente, lo que reduce aún más la superficie de ataque y minimiza cualquier configuración incorrecta que pueda generar problemas de seguridad.

Por supuesto, la necesidad de verificación no termina con la implementación. La autorización binaria también admite la validación continua, lo que permite el cumplimiento continuo de la política definida, incluso después de la implementación. Si una aplicación en ejecución no cumple con una política ya existente o nueva, se crea y registra una alerta, lo que garantiza que lo que se ejecute en producción sea exactamente lo que la empresa desea.

Administración de vulnerabilidades

Además de asegurar la integridad, otro aspecto de la seguridad de la cadena de suministro es garantizar que se encuentre con rapidez cualquier vulnerabilidad y se le aplique un parche. Los atacantes evolucionaron y ahora insertan de forma activa las vulnerabilidades en proyectos avanzados. La administración de vulnerabilidades y la detección de defectos debe estar incorporada en todas las etapas del ciclo de vida de la entrega de software.

Una vez que el código esté listo para su implementación, es posible usar una canalización de CI/CD y aprovechar las numerosas herramientas disponibles a fin de realizar un análisis completo del código fuente y los artefactos generados. Entre estas se encuentran analizadores estáticos, herramientas de fuzzing y varios tipos de herramientas para el análisis de vulnerabilidades. Después de implementar una carga de trabajo en producción, y mientras se ejecuta aquí y se entrega a los usuarios, es necesario supervisar las amenazas emergentes y contar con planes para poder solucionarlas al instante.

Conclusión

En resumen, para proteger una cadena de suministro de software es necesario implementar prácticas recomendadas, como SLSA, y usar servicios administrados confiables para lograr dicha implementación.

Es fundamental:

- Comenzar con el código y las dependencias de la propia organización, y asegurarse de que sean de confianza.
- Proteger el sistema de compilación y usar certificaciones para verificar el cumplimiento de todos los pasos de compilación necesarios.
- Garantizar que todos los paquetes y artefactos sean de confianza y que no se puedan manipular.
- Aplicar controles sobre quién puede implementar qué y mantener registros de auditoría, además de usar la autorización binaria para validar las certificaciones de cada artefacto que se desee implementar.
- Ejecutar las aplicaciones en un entorno de confianza y asegurarse de que nadie pueda manipularlas mientras se ejecutan, además de prestar atención a las vulnerabilidades recientes para poder proteger las implementaciones.

En Google, incorporamos las prácticas recomendadas en cada paso de este recorrido para nuestra cartera de productos, a fin de que los usuarios cuenten con una base confiable con la que puedan trabajar.

Capítulo

04 Primeros pasos

¿Todo listo para comenzar a proteger tu cadena de suministro de software? Para que quede claro, el punto de partida es bastante arbitrario: no existe ninguna acción que proteja toda tu cadena de suministro, y no hay una acción que sea más importante que cualquier otra en materia de seguridad. Dicho esto, a continuación te presentamos cuatro recomendaciones iniciales.

1. Aplica parches de software

Si implementaste un código en tu entorno de producción con vulnerabilidades conocidas, le has ahorrado trabajo a los atacantes. A partir de ese momento, no importa qué tan bien hayas protegido tu cadena de suministro de software porque ya habrán encontrado la forma de ingresar a ella. Por eso, la aplicación de parches es fundamental.

2. Toma el control de lo que se ejecuta en tu entorno

Una vez que completes la aplicación de parches, deberás controlar la cadena de suministro del software. Para comenzar, puedes confirmar que lo que ejecutas realmente proviene de tus herramientas de compilación o de repositorios de confianza. Ese nivel de control ayuda a evitar ataques intencionales y errores involuntarios, como es el caso de un desarrollador que implementó algo que no sabía que no era seguro. Esto proporciona una base sólida para agregar herramientas como pruebas de clics y la autorización binaria.

3. Garantiza la seguridad de los paquetes de proveedores externos

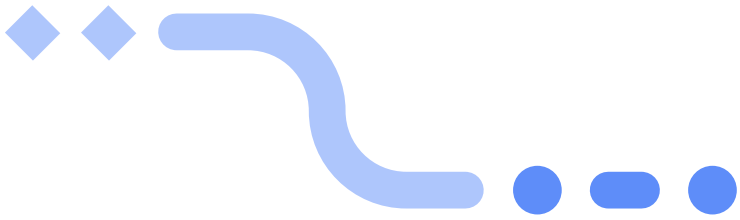
Un problema que emerge en la seguridad de la cadena de suministro es la frecuencia con la que el software de proveedores se ve comprometido y abre las puertas al ransomware o al acceso no autorizado a las implementaciones. Los paquetes de proveedores externos que ejecutas en tu entorno (por ejemplo, productos de administración de sistemas o redes, o productos de seguridad) suelen tener altos niveles de privilegio. Sugerimos pedirles a esos proveedores que vayan más allá de las declaraciones de seguridad estándares y proporcionen un nivel de certeza sobre los paquetes que usas. Puedes preguntarles cuál es su nivel de cumplimiento con SLSA o si cumplen con los requisitos del Decreto recientemente emitido..

4. Haz una copia privada de tu código fuente

Si usas software de código abierto, procura no usar una versión extraída directamente de Internet para compilar. En su lugar, realiza una copia privada que mantengas con parches a fin de comenzar de forma segura cada compilación y que puedas saber con un 100% de confianza de dónde proviene el código fuente.

Chapter

05 Recursos útiles



Prácticas recomendadas de DevOps

1. Seis informes anuales State of DevOps, un conjunto de artículos con información detallada sobre las capacidades que predicen el rendimiento en la entrega de software y una verificación rápida que te ayuda a descubrir en qué nivel te encuentras y cómo mejorar.
2. Informe Accelerate State of DevOps 2021 de Google Cloud
3. Informe de Google Cloud: Hacia una arquitectura nativa de la nube: un enfoque evolutivo sobre cómo aumentar la productividad de los desarrolladores a gran escala

Protección de la cadena de suministro de software

1. Blog de Google Cloud: ¿Qué es la seguridad de identidades de confianza cero?
2. Blog de seguridad de Google: Presentamos SLSA, un framework de extremo a extremo para la integridad de la cadena de suministro
3. Informe de Google Cloud: Desplazamiento hacia la izquierda en materia de seguridad: protección de la cadena de suministro de software

¿Todo listo para dar el siguiente paso?

Contáctanos para obtener más información sobre cómo Google Cloud puede ayudar a proteger tu cadena de suministro de software y tu negocio.

Comunícate con nosotros