
WHITEPAPER
Mitigando amenazas de seguridad en modelos serverless y FaaS
BELATRIX

Resumen ejecutivo

Dado que los productos y aplicaciones digitales están presentes en cada aspecto de la empresa moderna, la ciberseguridad es de suma importancia para las organizaciones; especialmente para aquellas que operan con productos web y arquitecturas en la nube.

Entretanto, con el auge de la tecnología serverless, las organizaciones están explorando nuevas ideas y conceptos acerca de la seguridad mientras transitan la siguiente etapa de desarrollo de software y gestión de productos.

En pocas palabras, la tecnología serverless permite que se ejecute la lógica de una aplicación sin que el equipo de tecnología tenga que preocuparse por administrar o mantener ningún servidor.

Con las aplicaciones serverless, solo se paga cuando se ejecutan funciones, lo que permite reducir costos.

A las empresas les resulta más fácil escalar y acelerar el lanzamiento de sus productos, ya que no es necesario administrar servidores ni sistemas operativos, ni preocuparse por actualizar hardware.

Todo esto se traduce en menos recursos de infraestructura y costos sustancialmente más bajos. Aun así, las amenazas de seguridad en esta tecnología son un desafío constante para muchas empresas que cuentan con una arquitectura de Function-as-a-Service (FaaS) basada en la nube.

Según un estudio de la empresa de seguridad Protego, el 98% de las funciones en aplicaciones serverless está en riesgo, y el 16% se encuentra en grave peligro.

El informe muestra que los errores de seguridad de mayor impacto son los permisos innecesarios, seguidos muy de cerca por código y configuraciones vulnerables.

Si bien estos problemas pueden mitigarse, puede resultar ser un proceso altamente desafiante si no se cuenta con las herramientas y los partners adecuados para dirigir el proceso.

En este reporte exploraremos algunas de las vulnerabilidades de seguridad más frecuentes en la tecnología serverless, explicaremos cuáles son los tipos de ataques y los accesos irregulares que suelen ocurrir, y ofreceremos asesoramiento sobre las mejores prácticas en seguridad.

We will examine:

- Common fallbacks and vulnerabilities
- 3 practical serverless security tips
- Belatrix's best practices and recommendations

Errores y vulnerabilidades frecuentes

En términos generales, las aplicaciones serverless son atacadas con el objetivo de dañar a una empresa, generalmente obteniendo acceso ilegal a datos confidenciales, que luego pueden ser saboteados, utilizados sin autorización e incluso vendidos en el mercado negro por los atacantes.

PureSec, una empresa de soluciones de seguridad, junto con varios de los mejores profesionales de la industria e investigadores de seguridad, definió una lista de las 10 vulnerabilidades más frecuentes y de mayor impacto para la tecnología serverless. A continuación enumeramos estas vulnerabilidades en orden de importancia:

- Inyección de datos de funciones de eventos
- Problemas de autenticación
- Configuración insegura de despliegues serverless
- Permisos y roles de función con privilegios

excesivos

- Monitoreo y registro de funciones inadecuados
- Dependencias de terceros inseguras
- Almacenamiento inseguro de secretos de aplicaciones
- Denegación de servicio y agotamiento de recursos financieros
- Manipulación del flujo de ejecución de funciones serverless
- Manejo inadecuado de excepciones y mensajes de error muy detallados

Analizaremos cada vulnerabilidad de la lista, observando por qué los atacantes podrían aprovecharlas y qué métodos pueden utilizar las empresas para prevenirlas.

Inyección de datos de funciones de eventos

Los error de inyección ocurren cuando los hackers logran engañar a una aplicación con datos poco confiables u hostiles para que ejecute comandos no deseados o dé acceso no autorizado a los datos confidenciales.

Una aplicación es más vulnerable a los ataques de inyección cuando no hay un protocolo de validación, filtración o saneamiento de los datos del usuario, lo cual

permite que los datos maliciosos puedan saltarse las defensas existentes.

Estos ataques pueden provocar una corrupción o pérdida de los datos, una prohibición de acceso o incluso una toma de control total del entorno de hosteo. Cuando el ataque es extremo, un hacker puede tomar el control de la ejecución de alto nivel de la aplicación y cambiar su flujo regular en un ataque de ransomware.

Traducir JavaScript Object Notation (JSON, por sus siglas en inglés) a texto sin formato de forma no segura podría generar vulnerabilidades de inyección. Esta función debe ser parte de la lógica principal de la aplicación para evitar accesos y ejecuciones de código arbitrarias, en cuyo caso muchos recursos serverless y en la nube podrían agotarse, como por ejemplo las direcciones IP y los límites de ejecución de funciones.

Una app es vulnerable a los ataques de inyección cuando no hay un protocolo de validación, filtración o saneamiento de los datos del usuario

Mantener los comandos y consultas por un lado y los datos por el otro es una buena práctica para evitar la

inyección. Es posible reducir estos ataques mediante el uso de prácticas de código seguro y el principio de mínimo privilegio (o PoLP, por su sigla en inglés).

Problemas de autenticación

Recomendamos que los recursos backend como bases de datos, eventos y funciones serverless, además de garantizar que la interfaz de usuario de la aplicación esté autenticando correctamente a los usuarios, no permitan el acceso a usuarios no autorizados.

Dado que las funciones serverless con privilegios excesivos son una vulnerabilidad de seguridad, es una buena práctica reducir lo más posible los privilegios de los recursos, lo que nuevamente se vincula con el principio de mínimo privilegio.

Las políticas de identidad pueden ser de ayuda con esto, habilitando o evitando a los usuarios realizar ciertas acciones en los recursos en la nube, como ejecutar funciones serverless o actualizar una tabla de una base de datos.

Los proveedores de servicios en la nube tienen condiciones de evaluación de políticas basadas en permisos configurables para los recursos que ofrecen.

Es sumamente recomendable revisar la lógica de evaluación de las políticas de identidad de un proveedor y utilizarla como punto de referencia para establecer políticas personalizadas que se apliquen a la arquitectura del software en particular. Un mal uso de estas políticas puede llevar a una exposición de los datos, que a su vez

puede llevar a que ocurran ataques de intermediarios basados en autorizaciones obtenidas de forma maliciosa o por contraseñas robadas.

Aplicar las mismas políticas y roles para todas las funciones, así como utilizar políticas que otorguen permisos sin restricciones para un recurso en la nube, no son buenas prácticas. Es más, comprender la forma en que los criminales se aprovechan de ciertos roles puede ser tan importante como tener un framework de seguridad sólido.

Configuración insegura de despliegues serverless

Los servicios en la nube y las arquitecturas serverless son ampliamente personalizables, debido a que responden a una variedad de necesidades, tareas o entornos específicos; pero esta flexibilidad puede tener un impacto significativo en la seguridad de una aplicación cuando esta última no está configurada de forma correcta.

La arquitectura serverless está orientada a hacer que las funciones sean stateless, lo cual hace que muchas aplicaciones dependan de la infraestructura de almacenamiento en la nube para almacenar datos entre ejecuciones. Cuando este almacenamiento no es seguro hay un mayor riesgo de exposición de datos corporativos confidenciales, especialmente cuando las autorizaciones y las autenticaciones están mal configuradas.

Mitigar estos riesgos es desafiante pero no imposible. Es recomendable tener conocimientos básicos sobre los controles de seguridad de almacenamiento que tienen

disponibles tus proveedores de servicios en la nube, como las configuraciones de almacenamiento, la autenticación multifactor y el cifrado de datos. También ayuda tener un equipo de desarrollo que comprenda plenamente las configuraciones de seguridad de la arquitectura serverless.

Permisos y roles de función con privilegios excesivos

En algunos casos, las aplicaciones serverless pueden contener cientos o miles de funciones, lo cual hace que la administración de permisos y roles sea una tarea tediosa y que requiera mucho tiempo. Para evitar esto, las organizaciones caen en la mala práctica de aplicar un único permiso o rol a todas las funciones, lo cual hace que cada función tenga acceso con privilegios excesivos a todo el sistema, aumentando los riesgos de seguridad.

Cada función serverless debe tener únicamente los permisos que necesita para realizar su lógica.

Cada función serverless debe tener únicamente los permisos que necesita para realizar su lógica: otra instancia del principio de mínimo privilegio. De otra

forma, los atacantes se concentrarán en funciones con privilegios excesivos para realizar tareas no autorizadas o perjudiciales.

Si bien asignar los permisos correctos a todas y cada una de las funciones en una aplicación serverless puede parecer una tarea demasiado larga, identificar permisos excesivos resulta vital para evitar ataques. Este proceso puede automatizarse de ser necesario.

Monitoreo y registro de funciones inadecuados

Algunas veces los atacantes prueban el comportamiento de la aplicación produciendo errores, por lo que recomendamos registrar información rastreable para detectar ataques.

Para los desarrolladores de aplicaciones serverless es una buena práctica implementar un mecanismo de auditoría que pueda analizar y eliminar registros periódicamente. Asegúrate de agregar registros tales como autenticación, eventos de cambio de datos y errores fatales a un sistema de gestión de información y eventos de seguridad (o SIEM, por su sigla en inglés). También se pueden guardar registros maliciosos para analizarlos.

Los registros son almacenados por los proveedores de servicios en la nube, y pueden estar en varios formatos (texto sin formato, JSON, etc). Las herramientas de administración como Logstash permiten a los desarrolladores visualizar y analizar la información contenida en los registros.

En general, se considera una mala práctica utilizar registros muy detallados en la etapa de producción cuando los registros no están correctamente asegurados, ya esto puede poner en riesgo de exposición a los datos confidenciales. Una alternativa es tokenizar la información antes de enviarla a un servicio de registro.

Dependencias de terceros inseguras

Sin importar cuán segura pueda parecer una aplicación serverless, la introducción de código de una dependencia de terceros insegura puede crear nuevos niveles de vulnerabilidad.

Siempre ten presente qué dependencias estás utilizando, así como los parches de seguridad que deberías instalar si hay un acceso irregular en la aplicación.

Como parte de un sólido proceso de integración continua y entrega continua, es importante mantener un inventario de los paquetes de software, dependencias y sus versiones en uso, y a su vez escanearlos en busca de vulnerabilidades frecuentes. Hay herramientas disponibles para actualizar las dependencias automáticamente, como Greenkeeper y NPM Audit.

Source: https://docs.npmjs.com/auditing-package-dependencies-for-security-vulnerabilities

Hacer una actualización a una nueva versión del paquete podría ser inseguro si un atacante ya está controlándolo y ha introducido un *malware*. Es vital usar solo paquetes de terceros confiables y asegurarse de que no estén infectados.

Desafortunadamente, decidir si se va a actualizar o no a una posible versión maliciosa es bastante engorroso y más difícil de automatizar. Es posible minimizar estas vulnerabilidades actualizando regularmente los paquetes a sus últimas versiones y eliminando dependencias innecesarias.

Almacenamiento inseguro de secretos de aplicaciones

El término "secretos" generalmente se refiere a claves o contraseñas, las cuales deben estar encriptadas de alguna manera. Los secretos se almacenan principalmente en una variable de entorno, en objetos S3 (blob), en bases de datos o en un administrador de secretos externo como HashiCorp Vault.

Si utilizas proveedores de servicios en la nube conocidos, como Amazon Web Services (AWS), Google Cloud Platform (GCP) o Microsoft Azure, éstos son responsables de la seguridad de esos secretos. Recomendamos tener presente dónde se almacenan los secretos durante el diseño de la arquitectura de la aplicación. En serverless, los secretos deben almacenarse únicamente en la nube.

Los expertos en seguridad consideran que es una mala práctica almacenar secretos en texto sin formato en un

En serverless, los secretos deben almacenarse únicamente en la nube.

repositorio de código o una variable de entorno, o intentar administrar demasiados secretos para un sistema. Intenta evitar el uso de configuraciones de timeout prolongadas, ya que esto puede llevar a alcanzar límites de ejecución o de carga, similar a lo que ocurre con los ataques de denegación de billetera (DoW, por su sigla en inglés).

Asegúrate de almacenar los secretos en un sistema de administración de secretos como Secrets Manager Service, que se integra con un servicio de administración de claves (KMS, por su sigla en inglés) para cifrar y descifrar secretos. Otra opción es Parameter Store de AWS, que viene completamente integrado con el KMS de Amazon. Los secretos también deben tener un tamaño limitado, restringirse a un pequeño grupo de personas, y registrarse en un sistema de auditoría utilizando un servicio como CloudTrail.

Por último, asegúrate de rotar frecuentemente los secretos, y cada vez que alguien abandone el equipo. Todo sistema de secretos también debe incluir una clave maestra cifrada que sea diferente para cada entorno activo y que rote regularmente. Estas prácticas deben formar parte de una estrategia de rotación de secretos y pueden automatizarse de ser necesario.

Denegación de servicio y agotamiento de recursos financieros

Los ataques de denegación de servicio (DoS, por su sigla en inglés) están diseñados para sobrecargar la capacidad de un servicio al obtener acceso a varios hosts, forzándolos a enviar una gran cantidad de solicitudes y potencialmente reteniendo a esos servicios como rehenes.

Estos ataques agotan el presupuesto de servicios de un cliente hasta el punto en que sus servicios ya no están disponibles; un resultado conocido como agotamiento de recursos financieros.

Agotar la capacidad de tiempo de ejecución reservada de una función puede conducir a un tipo de ataque DoS. Los atacantes también pueden establecer una dirección de falsificación obteniendo acceso no autorizado a un servicio secundario para sobrecargar la capacidad de los recursos y causar ataques de rescate. Otro tipo de ataque ocurre cuando los servicios de objetos exponen los datos del cliente al público por una configuración de permisos incorrecta.

Algunas opciones para prevenir este tipo de ataques DoS son la configuración adecuada de la autenticación requerida, establecer límites apropiados de timeout y de uso de disco para la ejecución de funciones serverless.

También podemos mencionar la implementación de firewalls de aplicaciones web de terceros de proveedores como F5 o Akamai, y la aplicación de servicios de protección para pipelines serverless, como AWS Shield,

GCP VPC Service o los controles de protección Azure DDoS.

Manipulación del flujo de ejecución de funciones serverless

Se sabe que algunos de los mayores impactos en las empresas son causados por ataques de lógica de negocio y manipulación de flujo, que son ataques muy complejos y difíciles de detectar.

Los atacantes intentan manipular el flujo de una aplicación con el objetivo de subvertir su lógica para eludir los controles de acceso, elevar los privilegios de usuario o incluso causar ataques DoS. En comparación con aplicaciones estándar, hay un mayor riesgo de estos ataques debido al diseño granular de las funciones serverless.

No existe una solución sencilla para este problema, pero es un buen comienzo establecer controles de acceso y permisos adecuados para cada función, y asegurarse de que los desarrolladores estén utilizando un servicio de administración de estado de aplicaciones si es necesario.

Manejo inadecuado de excepciones y mensajes de error detallados

En comparación con aplicaciones estándar, las opciones para depurar código de aplicaciones serverless línea por línea son limitadas y mucho más complejas, particularmente cuando se depura el código localmente. Estas limitaciones hacen que los desarrolladores opten

por programar mensajes de error que revelen información sensible o errores relacionados con el entorno, los usuarios o los datos de la aplicación, también conocidos como mensajes de error detallados (verbose error messages).

Para evitar el riesgo de accesos no autorizados a datos confidenciales, es una buena práctica que los desarrolladores utilicen las herramientas de depuración disponibles en la arquitectura serverless o las ofrecidas por el proveedor de servicios en la nube. También es una buena práctica definir mensajes de error personalizados que no revelen ningún detalle confidencial sobre el aplicación o la empresa.

Consejos prácticos de seguridad serverless

Además de abordar las errores y vulnerabilidades más frecuentes, nuestros equipos especializados en Belatrix han definido tres pasos prácticos clave que las organizaciones pueden seguir para proteger sus aplicaciones serverless.

1. Compartir la responsabilidad de seguridad con partners y proveedores

Cuando las empresas optan por una arquitectura serverless, los niveles de infraestructura son administrados al 100% por el proveedor de servicios en la nube, pero eso no significa que asuman el 100% de la responsabilidad de seguridad.

En la mayoría de los casos, es más beneficioso para ambas partes compartir la responsabilidad, y en otros casos es más ventajoso que el cliente sea totalmente responsable de ello. Por ejemplo, los proveedores de servicios en la nube siempre son responsables de los

entornos físicos de la tecnología serverless, como los centros de datos y todo el hardware necesario para ello.

El cliente es totalmente responsable de las regulaciones en materia de protección de datos y su cumplimiento, que deben estar alineadas en cualquier configuración serverless que estén operando. También debe administrar adecuadamente el acceso a los datos, ya que los proveedores de servicios en la nube solo pueden configurar permisos y privilegios en su extremo.

La instalación de parches de seguridad y actualizaciones tanto localmente como en la nube es una responsabilidad compartida del cliente y del proveedor, ya que el código malicioso puede afectar a los usuarios en cualquier extremo.

Los principales proveedores de la nube son AWS Lambda, Google Cloud Functions y Azure Function Apps. Todos ellos promueven un modelo de seguridad de responsabilidad compartida, así que asegúrate de que tus equipos lo entiendan y realmente tengan en cuenta sus responsabilidades acorde al modelo que aplique.

2. Mitigar ataques con DevOps

El uso de prácticas de DevOps como la integración continua y la entrega continua (CI y CD por sus siglas en inglés, respectivamente) en el ciclo de vida de desarrollo de software mejora en gran medida la seguridad de las aplicaciones serverless.

Las pipelines de aplicaciones FaaS automatizadas pueden ayudar a rastrear los flujos de datos más rápidamente, mitigando los ataques y permitiendo a los ingenieros tomar medidas rápidamente cuando surge una amenaza de seguridad. Una pipeline de DevOps automatizada reduce el riesgo de errores humanos, lo que resulta en aplicaciones serverless mucho más seguras.

El Centro de Excelencia de DevOps de Belatrix tiene una amplia experiencia trabajando con organizaciones para lograr este objetivo. Nuestro equipo crea pipelines automatizadas que utilizan las mejores prácticas de DevOps. Éstas incluyen el uso de herramientas de Infrastructure as Code (IaC) como Terraform y SAM (serverless application model), y herramientas de CI como GitLab, CircleCI o Github.

Las herramientas de IaC son útiles para la validación de seguridad tanto antes como después de la implementación. Combinadas con nuestra experiencia en testing y desarrollo, estas prácticas nos han permitido garantizar la seguridad de las soluciones FaaS durante todo el ciclo de vida de desarrollo de software.

Cada vez que se desarrollen nuevas funcionalidades o código para una aplicación serverless, tu equipo de tecnología debe aplicar las mejores prácticas de seguridad y luego revisarlas y probarlas, hasta que la pipeline se haya automatizado.

Así es como nos aseguramos de que se lancen al mercado productos de gama alta, con aplicaciones diseñadas para ser escalables y resistentes.

3. Asegurar la cadena de producción de aplicaciones

La cadena de producción de aplicaciones abarca todo lo que una aplicación necesita para llegar a la entrega del producto final. Esto incluye a los servicios de proveedores (como los servicios de seguridad de terceros), el tiempo de ejecución, los lenguajes de programación, el kit de desarrollo de software y las bibliotecas.

Recomendamos asegurarse de que se firmen acuerdos de nivel de servicio (ANS) con los proveedores que describan sus prácticas de seguridad y que establezcan en forma clara la cantidad de datos que administrarán. Otro aspecto a considerar son sus certificaciones de seguridad, como por ejemplo si cumplen con la ISO 27001 o con PCI.

Por último, asegúrate de que tu proveedor no utilice servicios, bibliotecas o paquetes que no tengan el mantenimiento al día. Esto también aplica si tu proveedor está utilizando bibliotecas con vulnerabilidades de seguridad sin parches.

Buenas prácticas y recomendaciones de Belatrix

En lo que concierne a la tecnología serverless, para las organizaciones exitosas la seguridad no es solo parte del desarrollo y el testing, sino también de las fases de diseño y arquitectura. Tener conocimiento sobre las vulnerabilidades de seguridad permite a las organizaciones prepararse mejor para los delitos cibernéticos y otras amenazas, y las ayuda a delinear mejor las pautas de seguridad como parte de su estrategia DevSecOps.

En todos los casos de desarrollo de aplicaciones serverless, hay prácticas de seguridad específicas e importantes que se deben seguir:

- Siempre realizar **revisiones de código**
- Actualizar los **tiempos de ejecución** de funciones serverless
- Mantener los **datos confidenciales cifrados**
- Restringir las cuentas del servidor al **mínimo**

privilegio posible

- **Administrar contraseñas** y autenticaciones correctamente

- Asegurarse de que todas las dependencias, incluyendo las bibliotecas de terceros, estén actualizadas y tengan las **últimas actualizaciones de seguridad**

- Controlar el acceso a funciones serverless usando **APIs seguras** u otros medios similares

- Utilizar las **mejores prácticas para aplicaciones serverless**, que cubrimos ampliamente en nuestro whitepaper "El Estado de la Tecnología Serveless"

- Si es posible, encarar los problemas de seguridad con **herramientas especializadas de terceros**

Es una buena práctica hacer un inventario de posibles objetivos que puedan resultar de interés para los atacantes. En general éstos suelen ser los datos del cliente o información sobre procesos empresariales confidenciales. Además, una vez que el software está en producción, es importante asegurarse de que las APIs no devuelvan más datos de los que deberían estar devolviendo. Esta es una clara alerta para una posible violación de seguridad.

Es una buena práctica hacer un inventario de posibles objetivos que puedan resultar de interés para los atacantes. En general éstos suelen ser los datos del

cliente o información sobre procesos empresariales confidenciales. Además, una vez que el software está en producción, es importante asegurarse de que las APIs no devuelvan más datos de los que deberían estar devolviendo. Esta es una clara alerta para una posible violación de seguridad.

Por lo general, no recomendamos crear un mecanismo de autenticación o autorización propio, ya que requiere esfuerzos de desarrollo adicionales y debe alcanzar un estándar de rendimiento muy alto. En su lugar, utiliza herramientas y estándares de autenticación existentes, como OAuth, que luego de numerosas pruebas han demostrado su buen funcionamiento en entornos de producción.

Por último, asegúrate de investigar en detalle a los proveedores de servicios en la nube y los proveedores de desarrollo de software para asegurarte de que su experiencia y sus conocimientos de seguridad serverless no sean deficientes. Probablemente las aplicaciones serverless y FaaS sean el futuro del software, por lo que colaborar con los protagonistas en el ámbito será clave para aprovechar nuevas oportunidades en la revolución serverless.

Acerca de Belatrix Software

Belatrix Software ayuda a las empresas a progresar en el mundo digital.

Las organizaciones se asocian con Belatrix para convertir ideas en productos de software innovadores y de alta calidad, en base a procesos perfeccionados de desarrollo Agile. Los clientes usan los servicios de transformación digital de Belatrix para crear productos de software superiores, reducir el tiempo de lanzamiento, y ganar ventajas competitivas.

Los laboratorios de Belatrix, enfocados UX, DevOps, desarrollo móvil, Inteligencia Artificial y QA Automation, ayudan a las empresas a convertirse en líderes digitales.

Los clientes de Belatrix incluyen tanto a empresas establecidas en los niveles de Fortune como a empresas emergentes respaldadas por capitales de riesgo. Algunos de los clientes de la firma son Disney, Adobe, AOL, PwC y Shutterfly.

Belatrix es una empresa sudamericana con oficinas en Florida, Nueva York, San Francisco, Mendoza, Buenos Aires, Bogotá y Lima.

Para conocer más, visita nuestro sitio: http://www.belatrixsf.com.

¡Contáctanos!