Documentación Técnica de Seguridad para Aplicaciones Django en Google Cloud

VML Data Science & Engineering Team

19 de febrero de 2025

Índice

1. Introducción

Este documento detalla las estrategias y acciones específicas implementadas para garantizar la seguridad de una aplicación web desarrollada con Django, desplegada en Google Cloud Platform (GCP) y utilizando Cloud SQL para el almacenamiento de datos. La aplicación está diseñada para ejecutarse en un entorno distribuido que incluye Cloud Run para endpoints sin servidor, Compute Engine para servicios adicionales y Cloud SQL como base de datos gestionada.

El objetivo principal de este documento es describir en detalle las medidas de seguridad adoptadas en cada capa de la arquitectura, incluyendo:

- La protección de endpoints en Cloud Run mediante autenticación segura, conexiones HTTPS y gestión de secretos.
- El almacenamiento seguro de datos en Cloud SQL utilizando encriptación, políticas de acceso y redes privadas.
- El despliegue seguro en Compute Engine, con control de accesos, actualización continua de parches y gestión de firewalls.

A diferencia de una guía genérica de buenas prácticas, este documento especifica las acciones concretas que se han implementado para mitigar riesgos de seguridad, así como las herramientas y servicios de Google Cloud utilizados para ello. Además, se incluyen pruebas de seguridad automatizadas (SAST y DAST), auditorías periódicas y políticas de seguridad continuas, como parte de un enfoque DevSecOps integral.

Este enfoque proactivo asegura que la aplicación no solo cumpla con los estándares de seguridad actuales, sino que también esté preparada para responder a amenazas emergentes en el entorno de la nube. La información proporcionada aquí está alineada con las mejores prácticas recomendadas por Google Cloud y el marco de trabajo de seguridad de Django, garantizando así una protección robusta y escalable.

2. Seguridad en Django

Django es un framework que prioriza la seguridad, proporcionando múltiples mecanismos integrados para proteger las aplicaciones web contra diversas amenazas. A continuación, se detallan las principales características y recomendaciones de seguridad que ofrece Django.

2.1. Protección contra Cross-Site Scripting (XSS)

Los ataques XSS permiten a usuarios malintencionados inyectar scripts en el navegador de otros usuarios. Django mitiga estos ataques mediante:

- **Escapado automático en plantillas**: El sistema de plantillas de Django escapa caracteres peligrosos de forma predeterminada, evitando la ejecución de scripts no autorizados. Sin embargo, es crucial tener precaución al utilizar `mark_safe` o desactivar `autoescape`, ya que pueden introducir vulnerabilidades si no se manejan correctamente.

2.2. Protección contra Cross-Site Request Forgery (CSRF)

Los ataques CSRF permiten a un atacante realizar acciones en nombre de un usuario autenticado sin su consentimiento. Django ofrece protección CSRF mediante:

- **Middleware de CSRF**: Activado por defecto en la configuración de `MIDDLEWARE`, este middleware verifica la presencia de un token secreto en cada solicitud POST, asegurando que las acciones sean intencionadas por el usuario.
- **Uso del tag `csrf_token`**: En las plantillas, es esencial incluir `{% csrf_token %}` dentro de cada formulario POST para generar el token necesario.

Para solicitudes AJAX, se debe configurar el encabezado `X-CSRFToken` con el valor del token CSRF obtenido de las cookies o del DOM.

2.3. Protección contra Inyección SQL

Django previene las inyecciones SQL al:

- **Uso de ORM**: El mapeador objeto-relacional (ORM) de Django construye consultas SQL utilizando parámetros, separando la lógica de la consulta de los datos proporcionados por el usuario, lo que evita la ejecución de código SQL arbitrario.

No obstante, al utilizar consultas SQL en bruto mediante `raw()` o `RawSQL`, es fundamental escapar y validar adecuadamente los parámetros para mantener la seguridad.

2.4. Protección contra Clickjacking

El clickjacking es un ataque donde un usuario es engañado para hacer clic en elementos ocultos de otro sitio. Django ofrece protección mediante:

- **Middleware de Clickjacking**: `XFrameOptionsMiddleware` añade la cabecera `X-Frame-Options` con el valor `DENY` o `SAMEORIGIN` a las respuestas HTTP, impidiendo que las páginas se carguen en iframes no autorizados.

2.5. Gestión Segura de Contraseñas

Para garantizar la seguridad de las contraseñas de los usuarios, Django implementa:

- **Almacenamiento seguro**: Las contraseñas se almacenan utilizando algoritmos de hashing robustos como `PBKDF2` de forma predeterminada. Es posible configurar otros algoritmos como `Argon2` o `bcrypt` mediante la opción `PASSWORD_HASHERS` en el archivo de configuración.
- **Validadores de contraseñas**: Django incluye validadores que imponen políticas de seguridad, como longitud mínima, prohibición de contraseñas comunes y restricciones en contraseñas numéricas. Estos se configuran en la opción `AUTH_PASSWORD_VALIDATORS`.

2.6. Middleware de Seguridad

Django proporciona un middleware específico para reforzar la seguridad de la aplicación:

- **SecurityMiddleware**: Este middleware permite configurar diversas políticas de seguridad HTTP, incluyendo:
  - **Redirección a HTTPS**: Mediante la opción `SECURE_SSL_REDIRECT`, se puede forzar la redirección de todas las solicitudes HTTP a HTTPS.
  - **Strict Transport Security (HSTS)**: Configurando `SECURE_HSTS_SECONDS`, se indica a los navegadores que deben comunicarse únicamente mediante HTTPS durante un período específico.
  - **Política de Referer**: La opción `SECURE_REFERRER_POLICY` controla la información de referencia enviada en las solicitudes, protegiendo la privacidad del usuario.

2.7. Validación de la Cabecera Host

Para prevenir ataques que explotan la cabecera `Host`, como envenenamiento de caché o CSRF, Django proporciona:

- **Configuración de `ALLOWED_HOSTS`**: Es esencial definir explícitamente los dominios permitidos en la opción `ALLOWED_HOSTS` del archivo de configuración, asegurando que solo las solicitudes dirigidas a estos dominios sean procesadas.

2.8. Prácticas Recomendadas Adicionales

Además de las protecciones integradas, se recomienda:

- **Uso de HTTPS**: Implementar HTTPS en todas las comunicaciones para proteger la integridad y confidencialidad de los datos transmitidos.
- **Gestión de contenido subido por usuarios**: Validar y sanitizar cualquier archivo o dato subido por los usuarios para evitar la ejecución de contenido malicioso.
- **Actualizaciones periódicas**: Mantener Django y sus dependencias actualizadas para beneficiarse de las últimas mejoras y parches de seguridad.

3. Seguridad en Google Cloud

Google Cloud Platform (GCP) ofrece una variedad de servicios que permiten desplegar y gestionar aplicaciones web de manera eficiente. Para garantizar la seguridad de una aplicación Django en este entorno, es fundamental considerar aspectos clave en los servicios utilizados, Cloud Run, Cloud SQL y Compute Engine.

3.1. Seguridad en Cloud Run

Cloud Run es un servicio de computación sin servidor que facilita el despliegue de aplicaciones en contenedores. Para asegurar los endpoints desplegados en Cloud Run, se recomienda:

- **Autenticación y Autorización**: Implementar mecanismos de autenticación para controlar el acceso a los servicios. Cloud Run permite configurar autenticación mediante identidades de servicio y, opcionalmente, integrar con Identity-Aware Proxy (IAP) para una capa adicional de seguridad.
- **Conexiones Seguras**: Asegurar que todas las comunicaciones se realicen a través de HTTPS. Cloud Run proporciona automáticamente un endpoint HTTPS para cada servicio, garantizando la encriptación de los datos en tránsito.
- **Variables de Entorno Seguras**: Evitar almacenar credenciales sensibles directamente en el código. Utilizar Secret Manager de GCP para gestionar y acceder a secretos de forma segura, y configurar las variables de entorno en Cloud Run para que obtengan estos valores de manera dinámica.

3.2. Seguridad en Cloud SQL

Cloud SQL es el servicio de bases de datos gestionadas de GCP. Para proteger los datos almacenados en una instancia de Cloud SQL, se establecen las siguientes prácticas:

- **Conexiones Seguras**: Utilizar el Cloud SQL Auth Proxy para establecer conexiones seguras entre la aplicación y la base de datos. Este proxy maneja la autenticación y encripta la comunicación, reduciendo la complejidad en la gestión de certificados SSL.
- **Gestión de Accesos**: Asignar roles y permisos mínimos necesarios a las identidades de servicio que acceden a la base de datos, siguiendo el principio de privilegio mínimo. Configurar las cuentas de usuario de la base de datos con contraseñas fuertes y rotarlas periódicamente.
- **Redes Privadas**: Implementar conexiones a través de IP privadas utilizando VPC (Virtual Private Cloud) para limitar la exposición de la base de datos a internet público. Esto se logra habilitando la comunicación privada entre los servicios de GCP y la instancia de Cloud SQL.

3.3. Despliegue Seguro en Compute Engine

Compute Engine permite desplegar máquinas virtuales (VMs) para ejecutar aplicaciones. Al desplegar una aplicación Django en Compute Engine, es esencial considerar:

- **Actualizaciones y Parches**: Mantener el sistema operativo y todos los paquetes actualizados para proteger la VM contra vulnerabilidades conocidas. Configurar actualizaciones automáticas cuando sea posible.
- **Firewall y Reglas de Red**: Definir reglas de firewall que restrinjan el acceso a la VM únicamente a las direcciones IP y puertos necesarios. Por ejemplo, permitir tráfico HTTP/HTTPS y bloquear otros puertos no utilizados.
- **Acceso Seguro a la VM**: Utilizar claves SSH para la autenticación y deshabilitar el acceso mediante contraseñas. Considerar el uso de herramientas como Google Cloud IAP para gestionar el acceso seguro a las VMs sin necesidad de exponerlas directamente a internet.
- **Almacenamiento de Secretos**: Evitar almacenar credenciales sensibles en el sistema de archivos de la VM. En su lugar, utilizar Secret Manager o almacenar variables de entorno de forma segura para que la aplicación acceda a ellas según sea necesario.

3.4. Acciones Específicas para Garantizar la Seguridad

Para asegurar la implementación de las mejores prácticas de seguridad en Google Cloud, se llevan a cabo las siguientes acciones:

- **Implementación de Monitoreo y Registro:** Se configura soluciones de monitoreo y logging utilizando Google Cloud Monitoring y Cloud Logging. Esto permite detectar actividades sospechosas, analizar registros de auditoría y responder rápidamente a posibles incidentes de seguridad.
- **Cifrado de Datos en Reposo y en Tránsito:**
  - Se habilita el cifrado de datos en reposo mediante las claves gestionadas por Google (Customer Managed Encryption Keys - CMEK) para Cloud SQL y Cloud Storage.
  - Se garantiza que todas las comunicaciones entre servicios utilicen HTTPS/TLS para proteger los datos en tránsito.
- **Revisiones de Seguridad y Pruebas de Penetración:** Se lleva a cabo auditorías de seguridad periódicas y pruebas de penetración utilizando herramientas de análisis estático (SAST) y dinámico (DAST). Estas pruebas se realizan antes de cada despliegue importante y como parte de un ciclo continuo de seguridad.
- **Despliegue de Políticas de Seguridad:**
  - Se aplican políticas de acceso con Identity and Access Management (IAM) para asegurar que solo las identidades de servicio autorizadas tengan acceso a recursos críticos.
  - Se configuran políticas de seguridad HTTP, incluyendo HTTP Strict Transport Security (HSTS) y Content Security Policy (CSP) para mitigar ataques XSS e interceptación de tráfico.
- **Gestión de Secretos:**
  - Se usa Google Cloud Secret Manager para almacenar y gestionar credenciales sensibles, tokens de acceso y variables de entorno críticas.

- Se configura el acceso a los secretos utilizando identidades de servicio con permisos estrictamente definidos.

- **Configuración de Firewalls y Redes:**
  - Se implementan reglas de firewall estrictas para permitir solo el tráfico necesario hacia Cloud Run, Compute Engine y Cloud SQL.
  - Se utilizan IPs privadas mediante VPC para asegurar la comunicación entre servicios sin exponerlos a internet público.

- **Automatización y DevSecOps:**
  - Se integran pipelines CI/CD con controles de seguridad para realizar análisis de vulnerabilidades en imágenes de contenedores y dependencias de código.
  - Se automatiza el despliegue seguro utilizando Infrastructure as Code (IaC) con herramientas como Terraform o Deployment Manager, asegurando que las configuraciones de seguridad sean consistentes en todos los entornos.

- **Capacitación y Concientización:**
  - Se realiza capacitación continua para el equipo de desarrollo en las mejores prácticas de seguridad en Google Cloud y Django.
  - Se establecen políticas internas de seguridad y respuesta a incidentes, asegurando que todos los miembros del equipo conozcan sus responsabilidades.

Estas acciones aseguran un enfoque proactivo y holístico en la protección de las aplicaciones Django desplegadas en Google Cloud, minimizando riesgos y fortaleciendo la postura de seguridad de la infraestructura.