Seguridad en Microservicios

Integrantes: Santiago Urrego – Santiago Gutiérrez – Luis Ramírez

1. Introducción a la Autenticación y Autorización en Microservicios

La arquitectura de microservicios ha revolucionado el diseño de aplicaciones distribuidas, permitiendo escalabilidad, resiliencia y modularidad. Sin embargo, este enfoque también incrementa los retos de seguridad, ya que cada servicio expone interfaces que pueden ser objeto de ataques. Dentro de este contexto, la autenticación y autorización son procesos fundamentales para garantizar que solo usuarios y servicios legítimos accedan a los recursos adecuados.

La autenticación consiste en verificar la identidad del usuario o servicio (¿quién eres?), mientras que la autorización se centra en determinar los privilegios asociados a esa identidad (¿qué puedes hacer?). En entornos distribuidos, estas funciones no deben implementarse de manera redundante en cada servicio, sino gestionarse de forma centralizada para asegurar consistencia y escalabilidad.

2. Uso de OAuth2 y OpenID Connect

OAuth2 es el estándar más utilizado para la autorización delegada en aplicaciones distribuidas. Permite que una aplicación (cliente) acceda a recursos protegidos en nombre del usuario sin necesidad de manejar credenciales directamente. OpenID Connect, construido sobre OAuth2, agrega una capa de autenticación, proporcionando información sobre la identidad del usuario mediante un ID Token firmado.

Conceptos clave incluyen:
- Access Token: credencial que permite acceder a recursos protegidos.
- Refresh Token: token de larga duración usado para renovar Access Tokens.
- Authorization Server: emite tokens tras autenticar al usuario.
- Resource Server: valida tokens y expone datos protegidos.

Este esquema es utilizado ampliamente por proveedores como Google, Microsoft y Facebook. En nuestra prueba de concepto, se integró el inicio de sesión con Google mediante OAuth2 y OpenID Connect, verificando la validez de los tokens en un backend desarrollado con FastAPI.

3. Seguridad en la Comunicación entre Microservicios: TLS y mTLS

La comunicación entre microservicios debe realizarse sobre canales seguros para prevenir ataques de interceptación (Man-in-the-Middle). Transport Layer Security (TLS) garantiza la confidencialidad e integridad de los datos transmitidos mediante cifrado. Sin embargo, en entornos altamente críticos se recomienda el uso de mutual TLS (mTLS), donde tanto el

cliente como el servidor presentan certificados válidos, estableciendo una autenticación bilateral.

El uso de mTLS en arquitecturas de microservicios es fundamental cuando se despliegan en entornos multi-tenant o expuestos a internet, asegurando que ningún servicio malicioso pueda hacerse pasar por otro legítimo.

4. Gestión de Secretos y Configuración Segura en Entornos Distribuidos

Uno de los errores más comunes en microservicios es la gestión insegura de secretos como claves API, contraseñas de bases de datos o certificados. Guardar estas credenciales en código fuente o archivos de configuración representa un riesgo significativo.

Buenas prácticas incluyen:
- Uso de gestores de secretos como HashiCorp Vault, AWS Secrets Manager o Kubernetes Secrets.
- Rotación periódica y automática de credenciales.
- Principio de privilegio mínimo: cada servicio debe tener acceso solo a los secretos que realmente necesita.
- Uso de variables de entorno seguras en lugar de valores hardcodeados.

5. Protección contra Ataques Comunes

Los microservicios incrementan la superficie de ataque, lo que obliga a considerar múltiples vectores de amenaza:

- Inyección SQL: mitigada mediante validación estricta de entradas y uso de ORMs seguros.
- Cross-Site Scripting (XSS): prevenido aplicando sanitización de datos, cabeceras CSP y escape de entradas.
- Cross-Site Request Forgery (CSRF): mitigado con tokens CSRF y configuraciones SameSite en cookies.
- Denegación de Servicio (DoS): limitado mediante rate limiting y control de tráfico en API Gateways.

La estrategia de seguridad Zero Trust cobra especial relevancia en este escenario: nunca confiar implícitamente en ningún servicio, incluso dentro de la red interna.

6. Prueba de Concepto

Como parte del trabajo práctico, se implementó una prueba de concepto de autenticación mediante Google OAuth2 con un frontend en HTML/JavaScript y un backend en FastAPI. El flujo consistió en:
1. El usuario selecciona su cuenta de Google en el frontend.
2. Google devuelve un ID Token firmado (JWT).
3. El backend valida el token con las librerías de Google y, si es válido, responde con la

información del usuario.

Este ejercicio demuestra cómo los microservicios pueden delegar la autenticación en un servidor de autorización externo (Google) y usar los tokens resultantes para proteger sus recursos. Se trata de una arquitectura extensible a entornos más complejos mediante API Gateways y servicios de identidad internos.

7. Conclusión

La seguridad en microservicios no puede ser tratada como un aspecto secundario, sino como un pilar fundamental de su diseño. Desde la autenticación federada con OAuth2/OpenID Connect, hasta la protección de la comunicación con TLS/mTLS y la gestión segura de secretos, cada componente debe abordarse con rigurosidad. La adopción de estrategias como Zero Trust y la integración de herramientas de observabilidad permiten fortalecer la resiliencia del sistema frente a amenazas reales.

Referencias

[1] OAuth 2.0 Framework - IETF RFC 6749. https://www.rfc-editor.org/rfc/rfc6749
[2] OpenID Connect Core Specification. https://openid.net/specs/openid-connect-core-1_0.html
[3] OWASP Top Ten Project. https://owasp.org/www-project-top-ten/
[4] NIST Special Publication 800-207: Zero Trust Architecture.
[5] HashiCorp Vault Documentation. https://developer.hashicorp.com/vault/docs