TALENTO TECH

TIC

REGIÓN 3

CAUCA – NARIÑO

LECCIÓN 1-UNIDAD 2

Universidad Tecnológica de Pereira

faceIT

Autenticación y autorización

TIC

A nivel de aplicación existen muchas soluciones de software que permiten que los datos sean enviados y compartidos para su gestión y visualización. Por ejemplo, aplicaciones que integran varios servicios, aplicaciones bancarias, aplicaciones de transporte y muchos otros casos, muestran cómo los datos de teléfonos móviles, cuentas de correo electrónico que recopilan datos personales y soluciones integradas comparten datos de usuarios y de corporaciones. Este tipo de integraciones automatiza mucho trabajo que antes era manual, sin embargo, es necesario estructurar la forma en que las aplicaciones se comunican y comparten datos para evitar fallas de seguridad.

La gestión segura de la identidad y el acceso se han convertido en pilares de la seguridad, por lo que se han desarrollado estándares para su manejo. Dos estándares comunes son OAuth 2.0 y OpenID Connect.

Estos estándares fueron creados para permitir la autorización y la autenticación de forma eficiente y segura, intentando facilitar las cosas para los usuarios y salvaguardar la información. Por ejemplo, cuando un servicio permite acceder a un usuario desde su identidad en redes sociales, sin crear un nuevo usuario y contraseña, se suele mejorar la experiencia para el usuario, evitando que recuerde nuevas contraseñas.

Protocolo OAuth 2.0

OAuth 2.0, que significa “Open Authorization” (autorización abierta), es un estándar diseñado para permitir que un sitio web o una aplicación accedan a recursos alojados por otras aplicaciones web en nombre de un usuario. OAuth 2.0 proporciona acceso consentido y restringe las acciones que la aplicación del cliente puede realizar en los recursos en nombre del usuario, sin compartir nunca las credenciales del usuario.

Se debe tener en cuenta que OAuth es un protocolo de autorización y no de autenticación. Es decir, permite a un aplicativo conceder acceso a grupos de recursos que le estén permitidos.

Para conceder el acceso se utilizan Tokens de acceso. Un Token de acceso es un dato que representa la autorización concedida al usuario para leer recursos. Estos Tokens pueden almacenarse en formato JSON Web Token (JWT). Con este formato los emisores de tokens de autorización pueden relacionar datos asociados junto con el token. Por ejemplo, el nivel del usuario, la fecha de caducidad del token y los recursos que puede ver y modificar.
Oauth 2.0 define roles para las autorizaciones:

Propietario de los recursos: Es el usuario o el sistema dueño de los recursos protegidos. Este nivel de usuario puede conceder acceso a los recursos que le pertenecen.

Cliente: Un cliente es un sistema que requiere acceso a un recurso protegido. Para acceder al recurso el cliente debe tener un Token de acceso generado por un servidor propietario de los recursos. En el token se puede validar el nivel de permisos que el usuario tiene.

Servidor de autorización: Es un sistema que recibe las solicitudes de tokens de acceso del cliente y las emite cuando el usuario propietario del recurse se ha autenticado y ha dado su consentimiento. El servidor de autorización expone dos puntos para la conexión. Un punto de conexión de autorización y un punto de conexión de toquen.

El punto de autorización maneja la autenticación interactiva y el consentimiento del usuario para entrar a un sistema. El punto de conexión de token permite validar las autorizaciones entre máquinas.

Servidores de recursos: Es un rol destinado a servidores que protegen los recursos. Este servidor recibe las solicitudes de acceso y se encarga de aceptar y validar los tokens que los clientes proporcionan. Cuando el token es validado, el servidor le permite ver los recursos que el usuario puede ver y modificar.

Tokens de Acceso y Código de Autorización en OAuth 2.0

OAuth 2.0 es un protocolo flexible diseñado para gestionar la autorización de forma segura entre diferentes aplicaciones y servicios. Un aspecto fundamental de este protocolo es cómo se manejan los tokens de acceso y el código de autorización. En lugar de otorgar directamente un token de acceso después de que el propietario del recurso autorice la solicitud, el servidor de autorización puede emitir un código de autorización, que el cliente intercambia más tarde por un token de acceso. Este enfoque mejora la seguridad, especialmente para aplicaciones sensibles como servicios bancarios o redes sociales.

Adicionalmente, el servidor puede generar un token de actualización junto con el token de acceso. Este token de actualización tiene una vida útil más prolongada y permite al cliente solicitar nuevos tokens de acceso cuando el original caducó. Dado su nivel de sensibilidad, los tokens de actualización deben ser almacenados de manera segura por el cliente, como en sistemas encriptados o almacenes de claves.

¿Cómo funciona OAuth 2?0?

Para utilizar OAuth 2.0, un cliente, como una aplicación móvil o un sitio web, primero necesita obtener sus credenciales (un ID de cliente y un client secret) del servidor de autorización. Estas credenciales identifican de forma única al cliente durante el proceso de autorización. A continuación, el cliente inicia una solicitud de autorización proporcionando estos datos junto con el alcance de los permisos requeridos (por ejemplo, "leer correos electrónicos") y una dirección de redirección para recibir la respuesta.

El proceso incluye varios pasos:

- El cliente solicita autorización: Proporciona sus credenciales y define los permisos requeridos.
- El servidor valida la solicitud: Comprueba si los permisos son válidos y autoriza la solicitud.
- El propietario del recurso concede acceso: Interactúa con el servidor para aprobar o denegar la solicitud.
- El servidor devuelve un código o token: Dependiendo del flujo, se entrega un código de autorización o un token de acceso.
- El cliente accede al recurso: Usa el token de acceso para interactuar con el servidor de recursos, como descargar un archivo o consultar un perfil de usuario.
- Por ejemplo, al iniciar sesión en una aplicación con tu cuenta de Google, el servidor de autorización de Google valida tu identidad y devuelve un token de acceso que la aplicación puede usar para obtener información de tu perfil.

Tipos de Concesión en OAuth 2.0

OAuth 2.0 ofrece diferentes flujos o concesiones para adaptarse a diversas situaciones. Algunos de los más comunes incluyen:

Código de autorización: El servidor emite un código de uso único, que luego se intercambia por un token de acceso. Este flujo es ideal para aplicaciones web, ya que el intercambio puede hacerse de forma segura en el backend. Por ejemplo, una tienda en línea puede usar este flujo para conectarse con una pasarela de pagos.

Flujo implícito: Aquí el token de acceso se devuelve directamente al cliente, simplificando el proceso. Sin embargo, esta concesión está obsoleta en muchos casos debido al riesgo de exposición del token en la URL.
PKCE (Proof Key for Code Exchange): Similar al código de autorización, pero añade medidas adicionales de seguridad para aplicaciones móviles o SPA (aplicaciones de una sola página). Por ejemplo, una aplicación de fitness podría usar este método para conectarse a una API de seguimiento de salud.

Credenciales del propietario del recurso: El cliente usa directamente las credenciales del usuario para obtener un token. Es adecuado solo para clientes de alta confianza, como aplicaciones internas de una empresa.

Credenciales de cliente: Usado en procesos automáticos, donde no hay interacción del usuario. Un ejemplo sería un servicio que sincroniza datos entre dos servidores.

Flujo de autorización de dispositivos: Diseñado para dispositivos con entrada limitada, como una Smart TV. El usuario ingresa un código en un navegador para autorizar el acceso.

Token de actualización: Permite obtener nuevos tokens de acceso sin requerir que el usuario vuelva a autorizar la aplicación. Por ejemplo, un cliente de correo electrónico puede usar este flujo para mantener el acceso a tu bandeja de entrada sin solicitar credenciales repetidamente. Cada tipo de concesión tiene sus ventajas según el caso de uso, garantizando seguridad y flexibilidad en diversas aplicaciones modernas.