Engineer

OAuth

dar acceso limitado a una aplicación sin darle la contraseña

Qué es

OAuth es un protocolo que permite a una aplicación utilizar los datos de una persona alojados en otro servicio, con su aprobación y dentro de unos límites que puede ver, sin que la aplicación llegue a conocer su contraseña. La versión en uso es OAuth 2.0, publicada por el IETF en octubre de 2012 como RFC 6749, y se encuentra escrita como OAuth 2.0 y OAuth2 con tanta frecuencia como simplemente OAuth. Sustituyó a OAuth 1.0, y ambas versiones son incompatibles.

La especificación distingue cuatro roles. El propietario del recurso es quien puede conceder acceso, normalmente una persona. El servidor de recursos contiene los datos protegidos y responde a peticiones que incluyen un token. El cliente es la aplicación que quiere acceder: una aplicación web, una aplicación de teléfono, un script o un agente de IA. El servidor de autorización inicia la sesión del propietario, le pide su consentimiento y emite tokens. Un producto pequeño puede ejecutar el servidor de recursos y el servidor de autorización como un único sistema; una empresa suele tener un proveedor de identidad delante de muchas API.

El cliente recibe un token de acceso, una cadena que representa un alcance y una duración de acceso concretos. Envía el token con cada petición al servidor de recursos. El RFC describe esa cadena como normalmente opaca para el cliente: el cliente la transporta, pero no está destinada a que la lea. El servidor de autorización también puede entregar un token de actualización, que el cliente canjea por un nuevo token de acceso cuando caduca el anterior, sin volver a preguntar a la persona.

Un cliente obtiene su primer token mediante una concesión. El RFC 6749 define cuatro: código de autorización, implícita, credenciales de contraseña del propietario del recurso y credenciales de cliente. La de credenciales de cliente sirve para un programa que actúa en su propio nombre, sin una persona involucrada. La concesión de código de autorización es la relevante para todo lo demás, porque las recomendaciones de seguridad actuales han retirado las concesiones implícita y de contraseña. Esas recomendaciones, junto con la prueba de clave para el intercambio de códigos (Proof Key for Code Exchange, PKCE) y las reglas de los tokens al portador, se están integrando en un único documento OAuth 2.1 del IETF. En su decimosexto borrador, fechado el 3 de septiembre de 2026, sigue siendo un Internet-Draft, un trabajo en curso y no un estándar.

OAuth trata de autorización, no de autenticación. Un token indica a la API lo que puede hacer el cliente; no le dice al cliente de forma fiable quién es la persona. Iniciar sesiones a partir de OAuth es la función de OpenID Connect, que añade un token destinado a que lo lea el cliente.

Utilice el concepto en una decisión

Aplique «¿Qué es OAuth?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Cuentas, contraseñas y permisos organizados para que cada persona tenga exactamente el acceso que necesita y lo pierda el mismo día que deja la empresa.

Explore nuestros servicios Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. 1.000 reglas RBAC para servidores Linux — 1,000 reglas de acceso escritas para cerrar una brecha de seguridad

1.000 reglas RBAC para servidores Linux Póngase en contacto →

Por qué importa

El RFC 6749 empieza por el problema que pretendía eliminar. Para permitir que una aplicación de terceros accediera a sus datos, usted le daba su contraseña. La aplicación tenía entonces que almacenarla, normalmente en texto claro. Podía acceder a todo lo que permitiera la contraseña y durante el tiempo que quisiera. Solo podía cortar el acceso de una aplicación cambiando la contraseña, lo que también cortaba el de todas las demás. Y cuando alguna sufría una intrusión, quedaban comprometidos la contraseña y todo lo que protegía. OAuth elimina cada uno de esos problemas: la aplicación nunca posee la contraseña, el token tiene un alcance y una caducidad, y puede retirarse el acceso de un cliente sin afectar a los demás.

Un equipo se encuentra con OAuth desde ambos lados. Cuando usted es responsable de una API, OAuth permite que socios, integraciones y herramientas de los propios clientes la llamen en nombre de un usuario sin que nadie almacene la contraseña de otra persona. Cuando consume una API, OAuth es lo que ofrecen calendarios, buzones, sistemas contables y almacenes de archivos, de modo que la parte cliente es código que su equipo escribe y mantiene. Los agentes de IA que actúan por una persona ocupan el mismo lugar que cualquier otro cliente: deberían tener un token limitado a lo que esa persona permitió, nunca sus credenciales.

Los alcances convierten el mínimo privilegio en algo que un servidor puede hacer cumplir. Un token que permita a un servicio de impresión leer fotos no debería permitirle borrarlas, y otro emitido para una API debería ser rechazado por otra. Las recomendaciones de seguridad del IETF de 2025, el RFC 9700, piden precisamente eso: limitar los privilegios de un token de acceso al mínimo necesario para el caso de uso y restringir los tokens al servidor de recursos para el que se emitieron.

Cómo se aplica en la práctica

El propio RFC ilustra el caso de un servicio de impresión que quiere obtener fotos de un sitio donde se comparten imágenes. Este es el intercambio mediante la concesión de código de autorización con PKCE, la forma que las recomendaciones actuales piden utilizar a todos los clientes. Antes de nada, el cliente genera un verificador de código aleatorio de entre 43 y 128 caracteres y lo guarda para sí. Calcula su hash con SHA-256 y redirige el navegador de la persona al servidor de autorización con el resultado codificado en base64url, el desafío de código:

```
GET /authorize?response_type=code&client_id=s6BhdRkqt3&state=xyz
    &redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
    &code_challenge=E9Melhoa2OwvFrEMTJguCHaoeK1t8URWbuGJSstw-cM
    &code_challenge_method=S256 HTTP/1.1
Host: server.example.com
```

El servidor de autorización inicia la sesión de la persona, muestra lo que solicita el servicio de impresión y, cuando esta lo aprueba, redirige el navegador de vuelta con un código de autorización de corta duración. El cliente comprueba que `state` tenga el valor que envió:

```
HTTP/1.1 302 Found
Location: https://client.example.com/cb?code=SplxlOBeZQQYbYS6WxSbIA
          &state=xyz
```

El código no vale nada por sí solo. El cliente lo canjea mediante una petición HTTPS al endpoint de tokens del servidor de autorización y esta vez envía el propio verificador:

```
POST /token HTTP/1.1
Host: server.example.com
Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&code=SplxlOBeZQQYbYS6WxSbIA
&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb
&client_id=s6BhdRkqt3
&code_verifier=dBjftJeZ4CVP-mB92K27uhbUJU1p1r_wW1gFWFOEjXk
```

El servidor calcula el hash del verificador, lo compara con el desafío que guardó junto al código y solo si coinciden devuelve los tokens, marcados para que ninguna caché los almacene:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
  "access_token":"2YotnFZFEjr1zCsicMWpAA",
  "token_type":"Bearer",
  "expires_in":3600,
  "refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA"
}
```

A partir de ahí, el cliente envía `Authorization: Bearer 2YotnFZFEjr1zCsicMWpAA` en cada llamada a la API de fotos. Una hora después, el token ha caducado, la API responde con `401 Unauthorized` y `error="invalid_token"` en su cabecera `WWW-Authenticate`, y el cliente canjea el token de actualización por uno nuevo. Pedir algo fuera del alcance concedido recibe, en cambio, `403 Forbidden` e `insufficient_scope`. El verificador protege contra el uso de un código interceptado: un código copiado del dispositivo durante su transmisión, el ataque

contra el que se diseñó inicialmente PKCE, no puede canjearlo nadie que carezca del verificador, y el propio verificador no se envió en la petición de autorización ni en la redirección del navegador.

Esa misma delegación permite que el software participe en flujos de trabajo con agentes. En la página de este sitio sobre convertir software empresarial existente en herramientas que puedan invocar agentes de IA, OAuth proporciona el acceso delegado: un agente actúa en nombre de un usuario sin intercambiar una contraseña y solo dentro de los permisos de ese usuario.

Dónde surgen los problemas

Una comprobación de redirección permisiva envía el código a otra persona. Un servidor de autorización que acepte cualquier URI de redirección que empiece por un prefijo registrado, o que coincida con un comodín, puede ser inducido a entregar un código a una página controlada por un atacante. El RFC 9700 exige que los servidores de autorización comparen las URI de redirección mediante coincidencia exacta de cadenas con las registradas, con una excepción para los números de puerto en localhost de aplicaciones nativas, y prohíbe tanto a clientes como a servidores utilizar redireccionadores abiertos que envíen el navegador a cualquier URL encontrada en un parámetro de consulta.

El callback acepta una respuesta que nadie solicitó. Si el cliente acepta cualquier código que llegue a su URI de redirección, un atacante puede introducir su propio código a través del navegador de una víctima y hacer que esta acabe trabajando en la cuenta del atacante. Es una falsificación de petición entre sitios dirigida a la redirección. Un cliente puede confiar en PKCE como protección una vez que haya confirmado que el servidor lo admite; de lo contrario, envía un valor `state` de un solo uso vinculado a la sesión del navegador de la persona y rechaza cualquier respuesta que no lo devuelva.

Los tokens viajan en las URL. La concesión implícita coloca el token de acceso directamente en la redirección, donde el historial del navegador y las cabeceras `Referer` pueden llevarlo a lugares a los que nunca debía llegar, y nada lo vincula al cliente que lo solicitó. El RFC 9700 dice que los clientes no deberían usarla, y el borrador OAuth 2.1 la omite. Lo mismo ocurre con una API que acepte un token como parámetro de consulta: el RFC 6750 dice que las URL de las páginas nunca deberían contener tokens al portador. Utilice la concesión de código y envíe los tokens en la cabecera `Authorization` .

La aplicación pide la contraseña de todos modos. La concesión de credenciales de contraseña del propietario del recurso permite que un cliente recopile el nombre de usuario y la contraseña de una persona y los cambie por un token, lo que anula la razón de ser de OAuth. El RFC 9700 establece que no debe usarse: expone las credenciales al cliente, acostumbra a las personas a escribir su contraseña en lugares distintos del servidor de autorización y nunca se diseñó para funcionar con autenticación de dos factores ni con inicios de sesión de varios pasos. Envíe a las personas al servidor de autorización con la concesión de código y utilice credenciales de cliente cuando no intervenga ninguna persona.

Un token robado funciona para cualquiera que lo posea. El RFC 6750 define un token al portador como aquel que puede utilizar cualquier parte que lo posea: tenerlo es toda la prueba. Utilice tokens de acceso de corta duración, restrinja cada uno a la API para la que se emitió y, cuando lo que está en juego lo justifique, vincule los tokens al cliente mediante TLS mutuo (RFC 8705) o DPoP (RFC 9449). Los tokens de actualización de clientes públicos, como las aplicaciones de teléfono y de navegador, deben estar vinculados al cliente o rotarse: cada actualización devuelve un token nuevo e invalida el antiguo. Si también se utiliza una copia robada, una de las partes acaba presentando un token invalidado, el servidor revoca el activo y el ataque se detiene a costa de que el cliente legítimo deba volver a pedir a la persona una nueva concesión.

Lecturas adicionales

- RFC 6749: marco de autorización OAuth 2.0 — los cuatro roles, los cuatro tipos de concesión y los endpoints, tal como se publicaron en octubre de 2012.
- RFC 7636: prueba de clave para el intercambio de códigos en clientes públicos de OAuth — verificador y desafío de código, con el ejemplo S256 utilizado arriba.
- RFC 9700: mejores prácticas actuales de seguridad de OAuth 2.0 — recomendaciones de enero de 2025: coincidencia exacta de redirecciones, PKCE para todos los clientes y eliminación de las concesiones implícita y de contraseña.
- RFC 6750: uso de tokens al portador — cómo presenta un cliente un token de acceso y cómo indica una API que falta, ha caducado o tiene un alcance demasiado limitado.
- Marco de autorización OAuth 2.1 — borrador del grupo de trabajo que integra estos documentos en uno y que en octubre de 2026 sigue siendo un Internet-Draft.

Glosario

Elija según sus restricciones reales

Describa la ruta de despliegue, los riesgos operativos y quiénes se encargan de las guardias. Podemos definir el alcance de un cambio con comprobaciones, una vía de recuperación y responsabilidades claras.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es OAuth?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

Explore nuestros servicios

welcome@engineer.company +45 50 10 61 84 Copenhague, Dinamarca

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/oauth/