Engineer

OpenID Connect

capa de identidad sobre OAuth para iniciar sesiones

Qué es

OpenID Connect es una forma estándar de que una aplicación averigüe quién es una persona a partir de un servicio que acaba de iniciar la sesión de esa persona. Su especificación lo llama una capa sencilla de identidad sobre OAuth 2.0, y así funciona: las mismas redirecciones, endpoints y tokens que OAuth, más un token adicional, un endpoint adicional y las reglas para comprobarlos. Suele abreviarse como OIDC. La especificación principal se hizo definitiva en febrero de 2014; el texto actual, que incorpora su segundo conjunto de correcciones, está fechado el 15 de diciembre de 2023 y lo mantiene un grupo de trabajo de la OpenID Foundation.

OAuth por sí solo responde a «¿qué tiene permitido hacer esta aplicación?». Le entrega un token de acceso destinado a una API, que no está pensado para que lo lea la propia aplicación. OpenID Connect añade «¿quién ha iniciado sesión?» y responde con un token de ID: un JSON Web Token dirigido a la propia aplicación que indica quién es la persona, quién avala esa identidad, cuándo se emitió y a qué aplicación está destinado. Los tokens de ID deben estar firmados; como excepción, la especificación permite `alg: none` únicamente si el tipo de respuesta no devuelve ningún token de ID desde el endpoint de autorización y el cliente solicitó explícitamente `none` al registrarse.

La especificación cambia los nombres de los roles de OAuth para esta función. El proveedor OpenID (OpenID Provider, OP) es un servidor de autorización OAuth que autentica a las personas y emite tokens de ID. La parte que confía en el proveedor (Relying Party, RP) es el cliente OAuth que necesita saber quién es la persona. La persona es el usuario final (End-User). Una petición se convierte en una petición OpenID Connect al incluir `openid` en su alcance, y ese valor es obligatorio: «Si no está presente el valor de alcance openid, el comportamiento queda completamente sin especificar».

Otros valores de alcance solicitan datos de perfil. `profile` abarca nombre, apodo, imagen, configuración regional y declaraciones similares; `email` añade `email` y `email_verified`, y `address` y `phone` hacen lo que indican sus nombres. La aplicación puede obtener esas declaraciones del endpoint UserInfo del proveedor, una API normal protegida por OAuth a la que se llama con el token de acceso.

Hay tres formas de utilizarlo. El flujo de código de autorización envía por el navegador solo un código de corta duración y obtiene los tokens mediante una petición HTTPS al endpoint de tokens; el flujo implícito devuelve los tokens directamente en la redirección; el híbrido combina ambos. Una especificación complementaria, OpenID Connect Discovery 1.0, permite a la aplicación consultar todos los endpoints y las claves de firma desde una única dirección.

Utilice el concepto en una decisión

Aplique «¿Qué es OpenID Connect?» 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

La razón para adoptarlo es el inicio de sesión único. Con un proveedor de identidad delante de muchas aplicaciones, las personas mantienen un solo conjunto de credenciales, las aplicaciones no almacenan ninguna y la política que decide cómo iniciar sesión, desde las reglas de contraseñas hasta un segundo factor, se establece una sola vez en el proveedor. Desactivar una cuenta en el proveedor impide que inicie una sesión nueva en cualquier aplicación que confíe en él, lo que convierte la gestión de bajas en un solo paso en vez de una lista.

También cierra una brecha de seguridad que puede aparecer cuando los equipos construyen el inicio de sesión sobre OAuth a secas. El RFC 6749 advierte de que, en el flujo implícito, un cliente público no tiene forma de saber a qué cliente se emitió un token de acceso, por lo que uno que suponga que solo el propietario del recurso puede entregarle un token válido queda expuesto a la suplantación:

un atacante que posea un token emitido a otra aplicación puede presentarlo y ser tomado por la persona que hay detrás. Un token de ID nombra la aplicación a la que está destinado en su declaración `aud`, y la aplicación la comprueba.

Al ser un estándar, el trabajo se puede trasladar. La especificación promete información de perfil interoperable, de modo que el código que inicia sesiones mediante un proveedor puede hacerlo mediante otro y, con Discovery, la aplicación descubre los endpoints y las claves de firma del proveedor a partir de su URL de emisor en vez de ajustes copiados a mano.

Por último, OpenID Connect da a cada persona un identificador estable. La especificación dice explícitamente que el par formado por `iss` —quién emitió el token— y `sub` —el identificador del sujeto— es el único par de declaraciones en el que una aplicación puede confiar como identificador estable de una persona. El valor de `sub` es único dentro del emisor, nunca se reasigna y tiene como máximo 255 caracteres ASCII. Un proveedor incluso puede emitir un `sub` distinto para cada aplicación, un identificador por pares, para que las aplicaciones no puedan correlacionar las actividades de una persona sin permiso.

Cómo se aplica en la práctica

Considere una aplicación web registrada en un OpenID Provider en `https://server.example.com` con el ID de cliente `s6BhdRkqt3`; los valores siguientes proceden de los propios ejemplos de la especificación. La aplicación empieza por leer la configuración del proveedor desde la dirección conocida bajo la URL del emisor:

```
GET /.well-known/openid-configuration HTTP/1.1
Host: server.example.com
```

La respuesta es un documento JSON. Una versión abreviada:

```
{
  "issuer": "https://server.example.com",
  "authorization_endpoint": "https://server.example.com/connect/authorize",
  "token_endpoint": "https://server.example.com/connect/token",
  "userinfo_endpoint": "https://server.example.com/connect/userinfo",
  "jwks_uri": "https://server.example.com/jwks.json",
```

```
  "scopes_supported": ["openid", "profile", "email", "address", "phone"],
  "id_token_signing_alg_values_supported": ["RS256", "ES256", "HS256"]
}
```

Discovery exige que el `issuer` de ese documento sea idéntico a la URL de emisor que la aplicación puso delante de la ruta conocida, y la aplicación se detiene si difiere. Después envía el navegador de la persona al endpoint de autorización con el flujo de código, el alcance `openid` y dos valores aleatorios que ha guardado vinculados a la sesión del navegador: `state` para la redirección y `nonce` para el token de ID. Un cliente público añadiría también los parámetros PKCE, que el RFC 9700 hace obligatorios para él; aquí se omiten para mantener cortas las líneas.

```
GET /connect/authorize?response_type=code&client_id=s6BhdRkqt3
    &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
    &scope=openid%20profile%20email
    &state=af0ifjsldkj&nonce=n-0S6_WzA2Mj HTTP/1.1
Host: server.example.com
```

El proveedor inicia la sesión de la persona, redirige el navegador de vuelta con un código y la aplicación canjea ese código en el endpoint de tokens exactamente igual que en OAuth a secas. La diferencia está en la respuesta, que ahora incluye un `id_token` junto al token de acceso —abreviado aquí—:

```
{
  "access_token": "SlAV32hkKG",
  "token_type": "Bearer",
  "refresh_token": "8xLOxBtZp8",
  "expires_in": 3600,
  "id_token": "eyJhbGciOiJSUzI1NiIsImtpZCI6IjFlOWdkazcifQ.ewogImlzc..."
}
```

El token de ID consta de tres partes base64url separadas por puntos. La primera se decodifica como `{"alg":"RS256","kid":"1e9gdk7"}` : el algoritmo de firma y el ID de la clave dentro del conjunto de claves publicado por el proveedor. La parte central contiene las declaraciones, y el ejemplo de la especificación dice:

```
{
  "iss": "https://server.example.com",
  "sub": "24400320",
```

```
  "aud": "s6BhdRkqt3",
  "nonce": "n-0S6_WzA2Mj",
  "exp": 1311281970,
  "iat": 1311280970,
  "auth_time": 1311280969,
  "acr": "urn:mace:incommon:iap:silver"
}
```

Antes de confiar en una sola palabra del contenido, la aplicación valida el token, y la especificación enumera los pasos. `iss` debe coincidir exactamente con el emisor del proveedor. `aud` debe contener el propio ID de cliente de la aplicación y ninguna audiencia en la que esta no confíe. La firma debe verificarse con las claves del proveedor obtenidas de `jwks_uri`, mediante el algoritmo de la cabecera; cuando el token llega directamente del endpoint de tokens por TLS, como aquí, comprobar el certificado TLS del servidor puede sustituir la comprobación de la firma. La hora actual debe ser anterior a `exp`. `nonce` debe ser igual al valor que envió la aplicación. Solo entonces inicia la sesión de la persona, usando como clave `https://server.example.com` junto con `24400320`, y utiliza el token de acceso para obtener declaraciones de correo y perfil del endpoint UserInfo si las necesita.

Un único proveedor para muchos sistemas supone un cambio tanto organizativo como técnico. Uno de los proyectos de este sitio, la gestión de acceso entre M365 e inicio de sesión único, consolidó identidades y accesos en M365, 1Password, Red Hat SSO y OKTA SSO, aplicó el mínimo privilegio y normalizó las altas y bajas para revocar el acceso con prontitud cuando alguien se marchaba.

Dónde surgen los problemas

Se usa el correo electrónico como clave de cuenta. Es la elección evidente y la equivocada. La especificación dice que un proveedor puede reutilizar una dirección de correo para otra persona con el tiempo y que `email`, `phone_number`, `preferred_username` y `name` no deben utilizarse como identificadores únicos. Una aplicación que busque personas por correo puede entregar la cuenta de alguien al siguiente propietario de esa dirección. Almacene `iss` y `sub` juntos como clave, conserve el correo como atributo y, antes de confiar en una dirección para cualquier uso, compruebe que `email_verified` sea true.

Un token de acceso se toma como prueba de quién inició sesión. Algunas aplicaciones reciben un token de acceso, llaman a una API con él y tratan la respuesta como un inicio de sesión. El token puede haberse emitido a una

aplicación completamente distinta; ese es el caso de suplantación sobre el que advierte el RFC 6749. Inicie la sesión con el token de ID y solo después de confirmar que su `aud` nombra su propio ID de cliente.

El token de ID se decodifica, pero nunca se comprueba. Leer las declaraciones de un token solo requiere decodificar base64url, sin ninguna clave, y esa facilidad es la trampa. Un token cuya firma, `iss`, `aud` o `exp` no se verifique puede ser falsificado, pertenecer a otra aplicación o haber caducado. Utilice una biblioteca mantenida, proporciónele el emisor y el ID de cliente y deje que ejecute todas las comprobaciones de la lista de la especificación, en lugar de elegir las cómodas.

El nonce falta, es fijo o nunca se compara. Sin él, un token de ID capturado de un inicio de sesión puede reutilizarse en otro. El RFC 9700 añade que el nonce debe ser específico de un inicio de sesión y estar vinculado al navegador que lo comenzó. La especificación sugiere una forma de hacerlo: guardar un valor criptográficamente aleatorio en una cookie de sesión HttpOnly, enviar su hash como `nonce` y compararlos cuando vuelva el token de ID.

Las claves de firma se guardan en caché para siempre. Los proveedores rotan las claves, y una aplicación que conserve el conjunto de ayer rechaza todos los tokens firmados con la nueva. La especificación espera que el proveedor publique las claves en `jwks_uri` y marque cada token con un `kid`, y que la aplicación vuelva a obtener el conjunto de claves cuando encuentre un `kid` que no conoce. Guarde las claves en caché, pero trate un `kid` desconocido como señal para actualizarla en vez de como motivo para hacer fallar el inicio de sesión.

Dos organizaciones cliente en un host se tratan como un único emisor. Un proveedor puede ejecutar varios emisores en un host, y la ruta forma parte del identificador del emisor: el sujeto `1234` en `https://example.com` y el sujeto `1234` en `https://example.com/sales` son personas distintas. Almacene la cadena completa del emisor, compárela exactamente y nunca la recorte hasta dejar solo un nombre de host.

Lecturas adicionales

- OpenID Connect Core 1.0 — token de ID, tres flujos, declaraciones estándar y pasos de validación en la edición que incorpora el conjunto 2 de erratas de diciembre de 2023.

- OpenID Connect Discovery 1.0 — documento de configuración de la dirección conocida y regla de que su emisor debe coincidir con la URL de emisor bajo la que se obtuvo.
- RFC 6749: marco de autorización OAuth 2.0 — marco sobre el que se construye OpenID Connect, con la advertencia de la sección 10.16 sobre usar tokens de acceso para identificar personas.
- RFC 7519: JSON Web Token (JWT) — formato de token utilizado por todos los tokens de ID.
- RFC 9700: mejores prácticas actuales de seguridad de OAuth 2.0 — recomendaciones actuales sobre PKCE, nonce y redirección, aplicables tanto a OpenID Connect como a OAuth.

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 OpenID Connect?» 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/openid-connect/