Engineer

OpenID Connect

Couche d'identité sur OAuth pour connecter les utilisateurs

Ce que c'est

OpenID Connect est un moyen standard pour une application d'apprendre qui est une personne auprès d'un service qui vient de l'authentifier. Sa spécification le décrit comme une simple couche d'identité au-dessus d'OAuth 2.0. Il fonctionne ainsi : les mêmes redirections, points de terminaison et jetons qu'OAuth, plus un jeton, un point de terminaison et les règles pour les vérifier. Son abréviation courante est OIDC. La spécification principale est devenue définitive en février 2014 ; le texte actuel, qui intègre son deuxième ensemble de corrections, date du 15 décembre 2023 et est maintenu par un groupe de travail de l'OpenID Foundation.

OAuth seul répond à « qu'est-ce que cette application a le droit de faire ? ». Il lui remet un jeton d'accès destiné à une API, que l'application n'est pas censée lire. OpenID Connect ajoute la question « qui s'est connecté ? » et y répond par un jeton d'identité : un JSON Web Token adressé à l'application elle-même, qui précise qui est la personne, qui en atteste, quand le jeton a été émis et à quelle application il est destiné. Les jetons d'identité doivent être signés. La seule exception autorisée par la spécification est `alg: none`, à condition que le type de réponse ne renvoie aucun jeton d'identité depuis le point de terminaison d'autorisation et que le client ait explicitement demandé `none` lors de son enregistrement.

La spécification renomme les rôles OAuth pour cet usage. Le fournisseur OpenID (OpenID Provider, OP) est un serveur d'autorisation OAuth qui authentifie les personnes et émet les jetons d'identité. La partie utilisatrice (Relying Party, RP) est le client OAuth qui a besoin de savoir qui est la personne. Celle-ci est l'utilisateur final (End-User). Une requête devient une requête OpenID Connect en incluant `openid` dans sa portée. Cette valeur est obligatoire : « Si la valeur de portée openid est absente, le comportement n'est pas du tout spécifié. »

D’autres valeurs de portée demandent des données de profil. `profile` couvre le nom, le pseudonyme, l’image, les paramètres régionaux et des déclarations similaires ; `email` ajoute `email` et `email_verified` ; `address` et `phone` correspondent à leur nom. L’application peut récupérer ces déclarations au point de terminaison UserInfo du fournisseur, une API ordinaire protégée par OAuth et appelée avec le jeton d’accès.

Il existe trois modes de fonctionnement. Le flux de code d’autorisation ne fait passer par le navigateur qu’un code de courte durée, puis récupère les jetons par une requête HTTPS au point de terminaison des jetons. Le flux implicite renvoie les jetons directement dans la redirection ; le flux hybride combine les deux. Une spécification complémentaire, OpenID Connect Discovery 1.0, permet à l’application de trouver tous les points de terminaison et toutes les clés de signature à partir d’une seule adresse.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce qu'OpenID Connect ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Des comptes, mots de passe et droits attribués selon les besoins exacts de chaque poste, puis retirés le jour du départ.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. 1 000 règles RBAC pour sécuriser Linux — 1,000 règles d'accès écrites pour combler une faille de sécurité

1 000 règles RBAC pour sécuriser Linux Contactez-nous →

Pourquoi c’est important

La raison de l’adopter est l’authentification unique. Avec un fournisseur d’identité devant de nombreuses applications, les personnes conservent un seul jeu d’identifiants, les applications n’en stockent aucun, et la politique de connexion, des règles de mot de passe au second facteur, se définit une seule fois chez le fournisseur. Désactiver un compte chez ce fournisseur l’empêche d’ouvrir une nouvelle session dans toute application qui lui fait confiance. Le retrait des accès devient ainsi une étape unique plutôt qu’une liste de tâches.

Il comble aussi une faille dans laquelle tombent les équipes qui construisent la connexion avec OAuth seul. La RFC 6749 avertit que, dans le flux implicite, un client public ne peut pas savoir pour quel client un jeton d’accès a été émis. Supposer que seul le propriétaire de la ressource peut lui remettre un jeton valide l’expose donc à l’usurpation : un attaquant possédant un jeton émis pour

une autre application peut le présenter et être pris pour la personne qui en est titulaire. Un jeton d'identité nomme son application destinataire dans sa déclaration `aud`, que l'application vérifie.

Comme il s'agit d'une norme, le travail se réutilise. La spécification promet des informations de profil interopérables : le code qui connecte les personnes par un fournisseur peut donc les connecter par un autre. Avec Discovery, l'application découvre les points de terminaison et les clés de signature du fournisseur à partir de son URL d'émetteur, plutôt que de réglages recopiés à la main.

Enfin, OpenID Connect donne à chaque personne un identifiant stable. La spécification précise que la paire `iss` (l'émetteur du jeton) et `sub` (l'identifiant du sujet) est la seule paire de déclarations qu'une application peut utiliser comme identifiant stable d'une personne. La valeur `sub` est unique chez l'émetteur, n'est jamais réattribuée et ne dépasse pas 255 caractères ASCII. Un fournisseur peut même émettre un `sub` différent pour chaque application, un identifiant propre à chaque paire, afin que les applications ne puissent pas corréler les activités d'une personne sans autorisation.

À quoi cela ressemble en pratique

Prenons une application web enregistrée auprès d'un fournisseur OpenID à `https://server.example.com` avec l'identifiant client `s6BhdRkqt3` ; les valeurs ci-dessous viennent des exemples de la spécification elle-même. L'application commence par lire la configuration du fournisseur à l'adresse normalisée de découverte située sous l'URL de l'émetteur :

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

La réponse est un document JSON. En voici une version abrégée :

```
{
  "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 l' `issuer` de ce document soit identique à l'URL d'émetteur que l'application a placée avant le chemin normalisé de découverte. L'application s'arrête s'ils diffèrent. Elle envoie ensuite le navigateur de la personne au point de terminaison d'autorisation avec le flux de code, la portée `openid` et deux valeurs aléatoires conservées en association avec la session du navigateur : `state` pour la redirection et `nonce` pour le jeton d'identité. Un client public ajouterait aussi les paramètres PKCE, que la RFC 9700 rend obligatoires pour lui ; ils sont omis ici pour garder des lignes courtes.

```
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
```

Le fournisseur authentifie la personne, renvoie le navigateur avec un code et l'application échange ce code au point de terminaison des jetons, exactement comme avec OAuth seul. La différence est dans la réponse, qui contient désormais un `id_token` à côté du jeton d'accès. En voici une version abrégée :

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

Le jeton d'identité comporte trois parties en base64url séparées par des points. La première se décode en `{"alg":"RS256","kid":"1e9gdk7"}` : l'algorithme de signature et l'identifiant de la clé dans le jeu de clés publié par le fournisseur. La partie centrale contient les déclarations. L'exemple de la spécification est le suivant :

```
{
  "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"
}
```

Avant d'accorder la moindre confiance au jeton, l'application le valide selon les étapes énumérées par la spécification. `iss` doit correspondre exactement à l'émetteur du fournisseur. `aud` doit contenir l'identifiant client de l'application et aucun destinataire auquel elle ne fait pas confiance. La vérification de la signature doit réussir avec les clés du fournisseur obtenues par `jwks_uri` , en utilisant l'algorithme de l'en-tête. Lorsque le jeton vient directement du point de terminaison des jetons par TLS, comme ici, la vérification du certificat TLS du serveur peut remplacer celle de la signature. L'heure actuelle doit précéder `exp` . Le `nonce` doit être égal à la valeur envoyée par l'application. Ce n'est qu'alors qu'elle connecte la personne, en utilisant `https://server.example.com` et `24400320` ensemble comme clé, et emploie le jeton d'accès pour récupérer les déclarations de courriel et de profil au point de terminaison UserInfo si elle en a besoin.

Un seul fournisseur pour plusieurs systèmes représente autant un changement organisationnel que technique. Un projet de ce site, la gestion des accès dans M365 et les systèmes SSO, a consolidé les identités et les accès dans M365, 1Password, Red Hat SSO et OKTA SSO, imposé le moindre privilège et standardisé les procédures d'arrivée et de départ afin que les accès soient rapidement révoqués lorsqu'une personne partait.

Où les problèmes surviennent

L'adresse de courriel sert de clé de compte. C'est le choix évident, et le mauvais. La spécification indique qu'un fournisseur peut réutiliser une adresse de courriel pour une autre personne au fil du temps et que `email` , `phone_number` , `preferred_username` et `name` ne doivent pas servir d'identifiants uniques. Une application qui retrouve les personnes par leur courriel peut remettre le compte de l'une au nouveau titulaire de cette adresse. Stockez `iss` et `sub` ensemble comme clé, gardez le courriel comme attribut et, avant de vous appuyer sur une adresse pour quoi que ce soit, vérifiez que `email_verified` est vrai.

Un jeton d'accès est pris pour une preuve de l'identité de la personne connectée. Certaines applications reçoivent un jeton d'accès, appellent une API avec lui et traitent la réponse comme une connexion. Le jeton a pourtant pu être émis pour une tout autre application : c'est l'usurpation contre laquelle la RFC 6749 met en garde. Connectez les personnes à partir du jeton d'identité, et seulement après avoir confirmé que son `aud` nomme votre propre identifiant client.

Le jeton d'identité est décodé mais jamais vérifié. Lire les déclarations d'un jeton ne demande qu'un décodage base64url, sans aucune clé : cette facilité est le piège. Un jeton dont la signature, `iss`, `aud` ou `exp` ne sont pas vérifiés peut être falsifié, appartenir à une autre application ou avoir expiré. Utilisez une bibliothèque maintenue, fournissez-lui l'émetteur et l'identifiant client, et laissez-la exécuter toutes les vérifications énumérées par la spécification au lieu de choisir les plus commodes.

Le nonce est absent, fixe ou jamais comparé. Sans lui, un jeton d'identité capturé lors d'une connexion peut être rejoué dans une autre. La RFC 9700 ajoute que le nonce doit être propre à une connexion et lié au navigateur qui l'a commencée. La spécification propose une méthode : conserver une valeur aléatoire de qualité cryptographique dans un cookie de session HttpOnly, envoyer son empreinte comme `nonce` et comparer les deux au retour du jeton d'identité.

Les clés de signature sont mises en cache pour toujours. Les fournisseurs renouvellent leurs clés, et une application qui garde le jeu de clés d'hier refuse tous les jetons signés avec la nouvelle. La spécification prévoit que le fournisseur publie les clés à `jwks_uri`, marque chaque jeton d'un `kid`, et que l'application récupère de nouveau les clés lorsqu'elle rencontre un `kid` inconnu. Mettez les clés en cache, mais traitez un `kid` inconnu comme un signal qu'il faut actualiser le jeu de clés, pas comme une raison de refuser la connexion.

Deux organisations clientes sur le même hôte sont traitées comme un seul émetteur. Un fournisseur peut exploiter plusieurs émetteurs sur un hôte, et le chemin fait partie de l'identifiant d'émetteur : le sujet `1234` chez `https://example.com` et le sujet `1234` chez `https://example.com/sales` sont des personnes différentes. Stockez la chaîne complète de l'émetteur, comparez-la exactement et ne la réduisez jamais à un nom d'hôte.

Pour aller plus loin

- OpenID Connect Core 1.0 — le jeton d'identité, les trois flux, les déclarations standard et les étapes de validation, dans l'édition intégrant le deuxième ensemble de corrections de décembre 2023.
- OpenID Connect Discovery 1.0 — le document de configuration à l'adresse normalisée de découverte et la règle exigeant que son émetteur corresponde à l'URL d'émetteur sous laquelle il a été récupéré.
- RFC 6749 : cadre d'autorisation OAuth 2.0 — le cadre sur lequel OpenID Connect repose, avec l'avertissement de la section 10.16 contre l'utilisation des jetons d'accès pour identifier les personnes.
- RFC 7519 : JSON Web Token (JWT) — le format utilisé par chaque jeton d'identité.
- RFC 9700 : bonnes pratiques actuelles pour la sécurité d'OAuth 2.0 — les recommandations actuelles sur PKCE, le nonce et la redirection, qui s'appliquent autant à OpenID Connect qu'à OAuth.

Glossaire

Choisissez selon vos contraintes réelles

Présentez le parcours de déploiement, les risques opérationnels et les personnes d'astreinte. Nous pouvons définir un changement avec des vérifications, une procédure de reprise et une responsabilité claire.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce qu'OpenID Connect ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

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

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

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

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/openid-connect/