Engineer

OAuth

✒ Accès limité à une appli sans partager votre mot de passe

Ce que c’est

OAuth est un protocole permettant à une application d’utiliser les données d’une personne détenues par un autre service, avec son accord et dans des limites qu’elle peut voir, sans jamais apprendre son mot de passe. La version utilisée est OAuth 2.0, publiée par l’IETF en octobre 2012 comme RFC 6749 ; OAuth 2.0 et OAuth2 se rencontrent aussi souvent qu’OAuth seul. Elle remplace OAuth 1.0 et les deux ne sont pas compatibles.

La spécification nomme quatre rôles. Le propriétaire des ressources peut accorder l’accès : c’est généralement une personne. Le serveur de ressources conserve les données protégées et répond aux requêtes portant un jeton. Le client est l’application souhaitant l’accès : application web ou mobile, script, agent IA. Le serveur d’autorisation connecte le propriétaire, demande son consentement et émet les jetons. Un petit produit peut réunir serveur de ressources et serveur d’autorisation dans un même système ; une entreprise place plus souvent un fournisseur d’identité devant de nombreuses API.

Le client reçoit un jeton d’accès, chaîne représentant un périmètre et une durée d’accès donnés. Il l’envoie avec chaque requête au serveur de ressources. La RFC dit que cette chaîne est généralement opaque pour le client : il la transporte, sans être censé la lire. Le serveur d’autorisation peut aussi émettre un jeton de renouvellement, échangé contre un nouveau jeton d’accès à l’expiration du précédent, sans solliciter à nouveau la personne.

Le premier jeton vient d’un mode d’octroi d’autorisation. La RFC 6749 en définit quatre : code d’autorisation, mode implicite, identifiants par mot de passe du propriétaire des ressources et identifiants du client. Ces derniers concernent un programme agissant pour lui-même, sans personne impliquée. Le code d’autorisation est le mode important pour le reste, car les recommandations de

sécurité actuelles ont abandonné les modes implicite et par mot de passe. Ces recommandations, Proof Key for Code Exchange ou PKCE et les règles des jetons au porteur sont en cours de regroupement dans un document OAuth 2.1 à l'IETF. À son seizième brouillon, daté du 3 septembre 2026, il reste un Internet-Draft, travail en cours et non une norme.

OAuth traite de l'autorisation, pas de l'authentification. Un jeton indique à l'API ce que le client peut faire ; il ne donne au client aucune information fiable sur l'identité de la personne. Authentifier les personnes sur la base d'OAuth relève d'OpenID Connect, qui ajoute un jeton destiné à être lu par le client.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce qu'OAuth ? » 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 RFC 6749 commence par le problème à résoudre. Pour laisser une application tierce atteindre vos données, vous lui donniez votre mot de passe. Elle devait le stocker, généralement en clair, et pouvait accéder à tout ce qu'il autorisait aussi longtemps qu'elle le voulait. Ne couper l'accès qu'à une application exigeait de changer le mot de passe, coupant aussi les autres. Si l'une était compromise, le mot de passe et tout ce qu'il protégeait l'étaient aussi. OAuth supprime ces problèmes : l'application ne détient jamais le mot de passe, le jeton a un périmètre et une expiration et l'accès d'un client peut être retiré sans toucher aux autres.

Une équipe rencontre OAuth des deux côtés. Si vous possédez une API, OAuth permet aux partenaires, intégrations et outils des clients de l'appeler au nom d'un utilisateur sans que quiconque conserve le mot de passe d'autrui. Si vous consommez une API, OAuth est ce que proposent calendriers, messageries, systèmes comptables et stockages de fichiers : la partie cliente est du code que votre équipe écrit et entretient. Les agents IA agissant pour une personne sont des clients comme les autres : ils devraient détenir un jeton limité à ce qu'elle a autorisé, jamais ses identifiants.

Les périmètres rendent le moindre privilège applicable par le serveur. Un jeton permettant à un service d'impression de lire des photos ne devrait pas permettre de les supprimer ; un jeton émis pour une API devrait être refusé par une autre. Les recommandations IETF de 2025, la RFC 9700, demandent précisément cela : privilèges d'un jeton limités au minimum nécessaire au cas d'usage et jetons restreints au serveur de ressources auquel ils sont destinés.

À quoi cela ressemble en pratique

L'illustration de la RFC est un service d'impression voulant des photos d'un site de partage. Voici cet échange avec code d'autorisation et PKCE, forme demandée pour tous les clients par les recommandations actuelles. Avant tout, le client crée un vérificateur de code aléatoire de 43 à 128 caractères qu'il garde pour lui. Il le hache avec SHA-256 et dirige le navigateur de la personne vers le serveur d'autorisation en joignant le résultat encodé en base64url, le défi de code :

```
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
```

Le serveur d'autorisation connecte la personne, montre la demande du service d'impression et, après son accord, renvoie le navigateur avec un code d'autorisation de courte durée. Le client vérifie que `state` correspond à la valeur envoyée :

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

Le code seul ne vaut rien. Le client l'échange par une requête HTTPS au point d'accès aux jetons du serveur d'autorisation, en envoyant cette fois le vérificateur lui-même :

```
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
```

Le serveur hache le vérificateur et le compare au défi conservé avec le code ; il ne renvoie les jetons qu'en cas de correspondance, avec des en-têtes interdisant leur conservation en cache :

```
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"
}
```

Le client envoie ensuite `Authorization: Bearer 2YotnFZFEjr1zCsicMWpAA` avec chaque appel à l'API des photos. Une heure plus tard, le jeton a expiré : l'API répond `401 Unauthorized` avec `error="invalid_token"` dans l'en-tête `WWW-Authenticate` et le client échange le jeton de renouvellement contre un nouveau jeton d'accès. Demander au-delà du périmètre accordé produit plutôt `403 Forbidden` et `insufficient_scope`. Le vérificateur protège contre l'utilisation d'un code intercepté : un code copié depuis l'appareil pendant son transfert, attaque initialement visée par PKCE, ne peut être échangé sans le vérificateur, et celui-ci n'a été envoyé ni dans la demande d'autorisation ni dans la redirection du navigateur.

Cette délégation permet aussi au logiciel de participer aux processus d'agents. Sur la page consacrée à transformer les logiciels métier existants en outils appelables par les agents IA, OAuth fournit l'accès délégué : un agent agit pour un utilisateur sans échange de mot de passe, et uniquement dans les limites de ses droits.

Où les problèmes surviennent

Un contrôle permissif des redirections livre le code à quelqu'un d'autre. Un serveur d'autorisation acceptant toute URI commençant par un préfixe enregistré ou correspondant à un joker peut être amené à livrer un code à une page de l'attaquant. La RFC 9700 exige une comparaison exacte des chaînes d'URI avec celles enregistrées, avec une exception pour les ports de localhost des applications natives, et interdit aux clients comme aux serveurs les redirections ouvertes vers n'importe quelle URL d'un paramètre de requête.

Le point de retour accepte une réponse jamais demandée. Si le client prend tout code reçu sur son URI de redirection, un attaquant peut envoyer son propre code par le navigateur de la victime : celle-ci travaille alors sur le compte de l'attaquant. C'est une falsification de requête intersite visant la redirection. Le client peut se fier à PKCE une fois qu'il a confirmé que le serveur prend PKCE en charge ; sinon, il envoie une valeur `state` à usage unique liée à la session du navigateur et refuse toute réponse ne la renvoyant pas.

Les jetons voyagent dans les URL. Le mode implicite place directement le jeton d'accès dans la redirection, d'où l'historique et les en-têtes `Referer` peuvent l'emporter ailleurs ; rien ne le lie au client demandeur. La RFC 9700 déconseille ce mode aux clients et le brouillon OAuth 2.1 l'omet. Il en va de même pour une API acceptant un jeton en paramètre de requête : la RFC 6750 recommande de ne jamais mettre de jeton au porteur dans une URL de page. Utilisez le code d'autorisation et envoyez les jetons dans l'en-tête `Authorization` .

L'application demande quand même le mot de passe. Le mode par identifiants du propriétaire des ressources permet au client de recueillir identifiant et mot de passe et de les échanger contre un jeton, annulant la raison d'être d'OAuth. La RFC 9700 interdit son utilisation : il expose les identifiants au client, habitue à saisir le mot de passe ailleurs que sur le serveur d'autorisation et n'a jamais été conçu pour l'authentification à deux facteurs ou les connexions en plusieurs étapes. Envoyez les personnes au serveur d'autorisation avec le code, et utilisez les identifiants du client si aucune personne n'intervient.

Un jeton volé fonctionne pour quiconque le détient. La RFC 6750 définit le jeton au porteur comme utilisable par toute partie qui le possède : sa détention est toute la preuve. Gardez des jetons d'accès à courte durée de vie, limitez chacun à l'API visée et, si l'enjeu le justifie, liez-les au client par TLS mutuel, RFC 8705, ou DPoP, RFC 9449. Les jetons de renouvellement des clients publics,

comme les applications mobiles et celles qui s'exécutent dans un navigateur, doivent être liés au client ou renouvelés par rotation : chaque renouvellement renvoie un nouveau jeton et invalide l'ancien. Si une copie volée est aussi utilisée, l'un des côtés présente un jeton invalidé ; le serveur révoque celui encore actif et l'attaque cesse, mais le client légitime doit redemander à la personne une nouvelle autorisation.

Pour aller plus loin

- RFC 6749 : cadre d'autorisation OAuth 2.0 — quatre rôles, quatre modes d'octroi et points d'accès, publiés en octobre 2012.
- RFC 7636 : PKCE pour les clients OAuth publics — vérificateur et défi de code, dont l'exemple S256 ci-dessus.
- RFC 9700 : bonnes pratiques actuelles de sécurité OAuth 2.0 — recommandations de janvier 2025 : redirections exactement comparées, PKCE pour chaque client et abandon des modes implicite et par mot de passe.
- RFC 6750 : utilisation des jetons au porteur — présentation d'un jeton par le client et réponse de l'API s'il manque, a expiré ou est trop limité.
- Cadre d'autorisation OAuth 2.1 — brouillon du groupe de travail réunissant ces textes, encore un Internet-Draft en octobre 2026.

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'OAuth ? » 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/oauth/