Engineer

OAuth

パスワードを渡さず、アプリに限られたアクセスを許す仕組み

どのようなものか

OAuthは、あるサービスにある個人のデータを、別のアプリケーションが、その人の許可を得て、本人が確認できる制限の範囲内で利用できるようにするプロトコルです。アプリケーションが本人のパスワードを知ることはありません。使われている版は、IETFが2012年10月にRFC 6749として公開したOAuth 2.0です。単にOAuthと書くのと同じくらい、OAuth 2.0やOAuth2という表記も見かけます。OAuth 1.0を置き換えたもので、両者に互換性はありません。

仕様は四つの役割を定めます。リソース所有者は、アクセスを許可できる主体で、通常は人です。リソースサーバーは、保護するデータを保持し、トークン付きのリクエストに応答します。クライアントは、アクセスしたいアプリケーションで、ウェブアプリ、携帯電話のアプリ、スクリプト、AIエージェントなどです。認可サーバーは、所有者をログインさせ、同意を求め、トークンを発行します。小さな製品では、リソースサーバーと認可サーバーを一つのシステムで動かせますが、企業では、一つのIDプロバイダーを多数のAPIの前に置く形がよくあります。

クライアントが受け取るのはアクセストークンです。一定のアクセス範囲と有効期間を表す文字列で、リソースサーバーへの各リクエストに付けます。RFCは、その文字列は通常、クライアントにとって不透明だと説明します。運びはしますが、読むためのものではありません。認可サーバーは、リフレッシュトークンも渡せます。アクセストークンが期限切れになると、クライアントは本人に再び尋ねず、それを新しいアクセストークンに交換します。

クライアントは、グラントという手段で最初のトークンを得ます。RFC 6749は、認可コード、インプリシット、リソース所有者のパスワード資格情報、クライアント資格情報という四つを定めます。クライアント資格情報は、人が介在せず、プログラムが自分自身として動く場合のものです。それ以外で重要なのは認可コードグラントです。現在のセキュリティー指針は、インプリシットとパスワ

ードのグラントを使わないものとしているからです。この指針と、コード交換のための鍵の証明であるProof Key for Code Exchange（PKCE）、Bearerトークンの規則は、IETFで一つのOAuth 2.1文書にまとめられつつあります。2026年9月3日付の第十六稿でも、まだInternet-Draftであり、作成途中の文書で、標準ではありません。

OAuthが扱うのは認可で、認証ではありません。トークンは、クライアントに許される操作をAPIに伝えますが、その人が誰かについて、クライアントに信頼できる情報を与えません。OAuthの上にログインの仕組みを設けるのは、OpenID Connectの仕事です。クライアントが読むためのトークンを追加します。

概念を判断に生かす

「OAuthとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 アカウント、パスワード、権限を整理し、業務に必要なアクセスだけを付与し、退職者の権限は当日に解除します。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 Linuxサーバーを守る1,000件のRBACルールを作成 — 1,000 セキュリティ上の穴を塞ぐアクセスルール

Linuxサーバーを守る1,000件のRBACルールを作成 お問い合わせ →

なぜ重要なのか

RFC 6749は、解消すべき問題から始まります。第三者のアプリケーションに自分のデータへのアクセスを許すには、パスワードを渡していました。アプリケーションは、それを通常、平文で保存する必要がありました。パスワードで届くすべての対象に、好きなだけ長くアクセスできました。一つのアプリケーションを締め出すには、パスワードを変えるしかなく、ほかのすべてのアプリケーションのアクセスも断たれました。どれか一つが侵害されれば、パスワードと、その背後のすべてが漏れました。OAuthは、それぞれを解消します。アプリケーションはパスワードを持たず、トークンに範囲と期限が付き、ほかに触れず、一つのクライアントのアクセスだけを取り消せます。

チームは、OAuthの両側に関わります。APIを提供する側では、パートナー、連携機能、顧客自身のツールが、ほかの人のパスワードを誰も保存せず、利用者に代わってAPIを呼べます。利用する側では、カレンダー、メールボックス、会計システム、ファイル保存サービスがOAuthを提供するため、チームはクライアント側のコードを書き、動作を維持します。人の代わりに動くAIエージェントも、ほかのクライアントと同じ立場です。その人が許した範囲だけのトークンを持つべきで、本人の資格情報は決して持つべきではありません。

スコープがあると、最小権限をサーバーが強制できます。写真の読み取りを印刷サービスに許すトークンで、写真の削除まで許すべきではありません。一つのAPI向けに発行したトークンは、別のAPIでは拒否すべきです。IETFの2025年のセキュリティー指針、RFC 9700は、まさにそれを求めます。アクセストークンの権限を、用途に必要な最小限に保ち、発行先のリソースサーバーに限定します。

実際の活用例

RFC自身の例は、写真共有サイトから写真を取得したい印刷サービスです。以下は、現在の指針がすべてのクライアントに求める、PKCE付きの認可コードグラントによる交換です。まずクライアントは、43から128文字のランダムなコードベリファイアを生成し、この段階では外へ渡さず手元に保持します。SHA-256でハッシュ化し、base64urlでエンコードした結果であるコードチャレンジを、本人のブラウザーとともに認可サーバーへ送ります。

```
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
```

認可サーバーは、本人をログインさせ、印刷サービスが求めるものを示します。本人が許可すると、短い有効期間の認可コードを付けてブラウザーを戻します。クライアントは、`state` が自分の送った値と一致するかを確かめます。

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

コード単独では役に立ちません。クライアントは、認可サーバーのトークンエンドポイントへのHTTPSリクエストでコードを引き換え、今度はベリファイアそのものを送ります。

```
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
```

サーバーは、ベリファイアをハッシュ化し、コードとともに保存したチャレンジと比較します。一致したときだけ、キャッシュに残さないよう指定したトークンを返します。

```
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"
}
```

以後、クライアントは、写真APIを呼ぶたびに `Authorization: Bearer 2YotnFZFEjr1zCsicMWpAA` を送ります。一時間後に期限が切れると、APIは `401 Unauthorized` を返し、 `error="invalid_token"` を `WWW-Authenticate` ヘッダーに含め、クライアントはリフレッシュトークンを新しいアクセストークンに交換します。許された範囲外を求めると、代わりに `403 Forbidden` と `insufficient_scope` が返ります。ベリファイアは、傍受されたコードの悪用を防ぐためのものです。PKCEが当初の対策対象とした、デバイスからの転送中にコピーされたコードは、ベリファイアを持たない者には引き換えられません。ベリファイア自体は、認可リクエストにもブラウザーのリダイレクトにも送っていません。

同じ委譲の仕組みによって、ソフトウェアはエージェントのワークフローにも参加できます。このサイトの、既存の業務ソフトウェアを、AIエージェントが呼び出せるツールに変えるページでは、OAuthが委譲されたアクセスを提供します。エージェントは、パスワードを受け渡さず、利用者の権限の範囲内だけで、その人の代わりに動きます。

問題が起きるところ

緩いリダイレクト検査で、コードが他人に届きます。 登録したプレフィックスで始まるURIや、ワイルドカードに一致するURIなら何でも受け入れる認可サーバーは、攻撃者のページへコードを送るよう誘導されます。RFC 9700は、登録

したURIとの完全な文字列一致で、リダイレクトURIを照合するよう求めます。例外は、ネイティブアプリのlocalhostのポート番号だけです。また、クエリーパラメーターにある任意のURLへブラウザーを送るオープンリダイレクターを、クライアントにもサーバーにも禁じます。

コールバックが、誰も求めていない応答を受け入れます。 リダイレクトURIに届いたコードを何でも受け入れると、攻撃者は、被害者のブラウザーを通して自分のコードを送れます。被害者は、攻撃者のアカウントで作業することになります。リダイレクトを狙うクロスサイトリクエストフォージェリです。サーバーのPKCE対応を確認できれば、クライアントはPKCEを防御に使えます。それ以外は、本人のブラウザーセッションに結び付けた一回限りの `state` 値を送り、その値を返さない応答を拒否します。

トークンがURLで運ばれます。 インプリシットグラントは、アクセストークンを直接リダイレクトに入れます。ブラウザーの履歴や `Referer` ヘッダーによって、想定外の場所へ漏れる場合があり、要求したクライアントとの結び付きもありません。RFC 9700は、クライアントは使うべきでないとし、OAuth 2.1の草案も、このグラントを含めません。トークンをクエリーパラメーターで受け取るAPIも同じです。RFC 6750は、ページのURLでBearerトークンを決して運ぶべきではないとします。認可コードグラントを使い、トークンは `Authorization` ヘッダーで送ります。

結局、アプリがパスワードを求めます。 リソース所有者のパスワード資格情報グラントでは、クライアントが本人のユーザー名とパスワードを集め、トークンに交換できます。OAuthの存在理由を失わせます。RFC 9700は、使用を禁止します。資格情報をクライアントにさらし、認可サーバー以外にもパスワードを入力する習慣を付け、二要素認証や複数段階のログイン向けには、最初から設計されていないからです。本人を認可コードグラントで認可サーバーへ送り、人が全く介在しないときは、クライアント資格情報を使います。

盗まれたトークンは、持っている人が誰でも使えます。 RFC 6750のBearerトークンは、所有するどの主体も使えるものです。持っていることが、証明のすべてです。アクセストークンの有効期間を短くし、発行先のAPIだけに制限します。リスクに見合う場合は、相互TLS（RFC 8705）やDPoP（RFC 9449）で、トークンをクライアントに結び付けます。携帯電話やブラウザーのアプリのようなパブリッククライアントのリフレッシュトークンは、クライアントへの結び付けか、ローテーションが必須です。ローテーションでは、更新のたびに新しいトークンを返し、古いものを無効にします。盗んだコピーも使われると、片方が無効なト

ークンを提示することになり、サーバーが有効なトークンを取り消して、攻撃が止まります。その代償として、正当なクライアントは、本人に新たなグラントを求める必要があります。

参考資料

- RFC 6749：OAuth 2.0認可フレームワーク — 2012年10月に公開された、四つの役割、四種類のグラント、エンドポイントです。
- RFC 7636：OAuthパブリッククライアントのコード交換のための鍵の証明 — コードベリファイアとチャレンジ、および上で使ったS256の例です。
- RFC 9700：OAuth 2.0の現在の最良セキュリティー実務 — 2025年1月の指針です。リダイレクトの完全一致、すべてのクライアントでのPKCE、インプリシットとパスワードグラントの不使用を示します。
- RFC 6750：Bearerトークンの使用 — アクセストークンの提示方法と、欠落、期限切れ、範囲不足をAPIが伝える方法です。
- OAuth 2.1認可フレームワーク — 関連文書を一つにまとめる作業部会の草案で、2026年10月時点でもInternet-Draftです。

用語集

実際の制約に合わせて選ぶ

デプロイの流れ、運用上のリスク、緊急対応の担当者をお知らせください。検査、復旧手段、明確な責任の所在を伴う変更の範囲を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「OAuthとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

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

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/oauth/