title: OAuthを理解しよう tags: oauth2 Web アクセストークン author: cotton*alta*

slide: false

こちらのページはTokyo City University Advent Calendar 2019の17日目の記事です。昨日の記事は杉田玄白さんの「TCUバードって知ってる？」でした。

今回node.jsのコラム的なものを書きたいと予定していたのですが時間が足りずコミケで使用した記事をQiitaに転載する形で参加させていただきます。

OAuthに対する正しい認識と用法を知ろう

最近流行りのOAuthなのですが、この仕組みについて間違えた理解をしている人が多いかと思います。例えばOAuth認証というワード、こちらは誤解の元凶ですね。使い方的には間違えてないんですがこれだけだとOAuthはTwitterなどの他サービスを用いたログインのための認証システムと勘違いされてもおかしくない気もします。（Twitter等のSNSを用いたログインでは基本的なユーザ情報にアクセスすることをOAuthで許可することによって許可され取得したデータをログインのために使うのであってログイン処理はOAuthではありません。）OAuthとは、リソースオーナー（利用者）が同意することによってクライアント（アプリなど）が限定的な権限でリソースサーバーにあるリソースオーナーのデータを使用できるようにするための仕組みのことです。クライアントは自分の権限を代理で使用するいわゆる弁護人というようなイメージです。

各種用語の解説

OAuthの具体的な流れを説明する前に前準備として理解しておくべき用語がありますのでここで紹介させていただきます。

トークンの種類

- アクセストークン クライアントがデータを利用する権限を持っていることを証明するもの。認可サーバーから要求されたスコープに応じたものが発行されます。アクセストークンは流出を防ぐために後述するリフレッシュトークンより有効期限が短くなっています。
- リフレッシュトークン アクセストークンの有効期限が切れた際、リフレッシュトークンを用いると再びアクセストークンを発行することができます。
- エンドポイント 各要素から発行されるURIのことです。以下の種類があります。
- 認可エンドポイント 認可サーバーが与えるURIです。クライアントからの要求に対してリソースオーナーの同意が行われたことを示します。
- トークンエンドポイント 認可サーバーが与えるURIです。このURIにアクセスすることによりクライアントはアクセストークンを取得することができます。

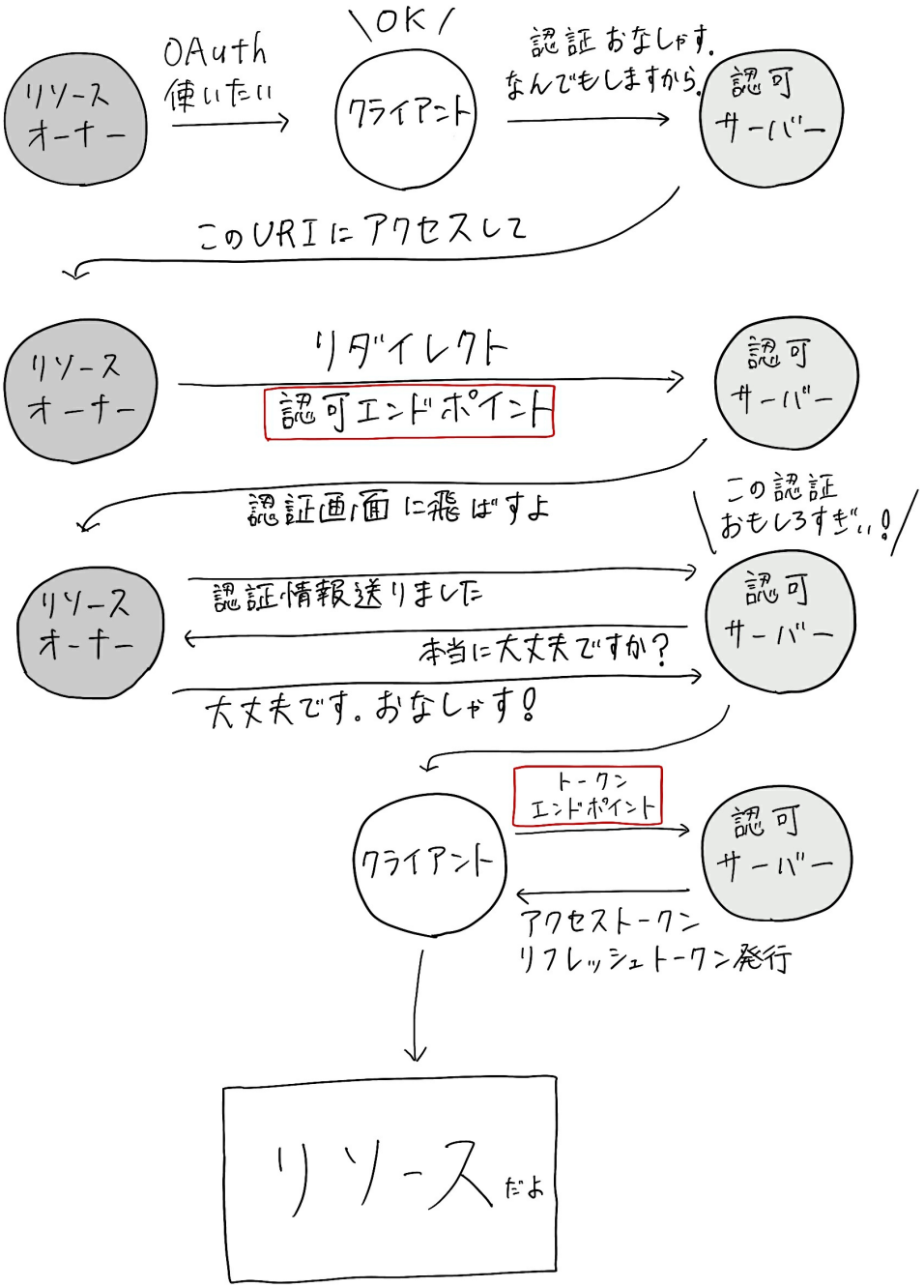
色んなタイプありますけど割愛させていただきます

OAuthと言っても一口縄ではいきません。グラントタイプよってさまざまな種類に分かれています。グラントとは付与タイプのこと、要するにグラントタイプによって権限を付与する流れが変わってきます。全部理解しようとすると骨が折れますので今回は主要なグラントタイプだけ理解しちゃいましょう。（実際のところそれぞれのタイプ間では細かな差があるくらいで大まかな流れは同じです。）

認証コードグラント

このタイプでは、リソースオーナー、クライアント（いわゆるアプリやwebサイト）、認証サーバー、リソースサーバーの4つの

要素が出てきます。このグラントタイプはクライアントと認可サーバーの間で認証が行われるのでセキュアであるところがメリットとして挙げられます。セキュアであるためアクセストークンとリフレッシュトークンの両者を発行することができます。図で見た方が早いので図にしてみました。全体図は次の図のようになっています。（突貫だったので手書きです、申し訳ありません）



流れをかみ砕く

まず最初にリソースオーナーがクライアント、要するにアプリにアクセスします。この時、リソースオーナーはそのクライアントのサービスにパスワードなど機密情報を預けるのが不安、もしくは新しくアカウントを作るのが面倒と思うかもしれませ

ん。クライアント側もリソースにはアクセスしたいがパスワードなど漏れるとマズイ情報を預かりたくないと思うこともあるかもしれません。そういう場合にOAuthを使うと、機密情報をリソースオーナーとクライアント間で直接受け渡しする必要なく、クライアントしか扱えないリソースをリソースオーナーの代わり、いわゆる代理で扱うことができるようになります。代理でリソースを扱うために、クライアントはリソースオーナーからの要求に対して認可サーバーにアクセスするためのリダイレクトURIを発行します。このURIがいわゆる認可エンドポイントです。これは今からあなたのリソースを代理で使いますよ？という事前確認を行うためのものです。リソースオーナーは認可サーバーにアクセスし、クライアントが使用したい権限の利用に対して認証情報を送る等の過程を経て同意を行います。認可サーバーは認証を経たリソースオーナーからアクセスを受けるとトークンエンドポイントを発行し、クライアントに返します。このURIにクライアントがアクセスするとようやくアクセストークンとリフレッシュトークンが発行されます。クライアントはこのアクセストークンを利用し許可されたリソースにアクセスしていくことになります。簡単な例を挙げるとすればTwitterの診断メーカーがあります。これは必要な権限としてユーザの基本的な情報とツイートを代理で行う権限を要求します。この要求する権限はクライアント側で自由に設定できるため、どのような権限を要求しているのか同意を行う際しっかりと確認しなければいけません。さもなければ意図してない情報の流出を許してしまうことになります。実際に必要以上の権限を要求してくるクライアントは多数存在してますので日頃から注意しましょう。

その他のグラントタイプ

- インプリシットグラント 現在非推奨となっているグラントタイプです。非推奨な理由としては認証の段階でクライアントに直接アクセストークンが渡されるため流出リスクが高いためとなっています。
- クライアントクレデンシャルグラント クライアントとリソースオーナーは通常分かれていますが、このグラントタイプではクライアントがリソースオーナーも兼ねることになっています。
- リソースオーナーパスワードクレデンシャルグラント ユーザーがIDとパスワードをクライアントへ送信するとそれ以降の処理はクライアントと認証サーバーで完結するタイプです。

おわりに

以上になります。実践的な内容ではないのですが仕組みの理解はできましたか？実践経験を積みたい方はQiita等で調べてみると無限に記事がでてくるのでそちらを参考にしてみてはいかがでしょうか。拙い文章で分かりにくかったかと思いますが、最後まで読んでくださった方ありがとうございました。もし間違い等あればコメント欄にてコメントお願いします。

次の記事はナカXトさんです。よろしくお願いします。