OAuth 2.0とOpenID Connect入門

yamotonalds

2017-01-30

OAuth 2.0とOpenID Connect入門

なぜこの内容にしたか

GCSにファイルをアップロードしているところで

invalid_grant: Invalid JWT: Token must be a short-lived token and in a reasonable timeframe

原因としてはサーバーの時刻ずれ

Google OAuth2.0 認証時に JWT::InvalidIatError: Invalid iat エラーが頻発する@NOTE

これを期に認証周りの仕組みを再確認しておこう

ちなみに

JWTはジョットと発音するそうです。

https://tools.ietf.org/html/rfc7519#section-1

OAuth 2.0とOpenID Connect

Google APIの認証・認可

認可にOAuth 2.0、認証にOpenID Connectを使っている

https://developers.google.com/identity/protocols/OAu

OAuth 2.0とは

ユーザーがアプリケーションにID・パスワードを教えることなく、アプリケーションがユーザーの代わりにユーザーデータにアクセスできるようにする仕組み

安全に 認可を取得するための認可プロセスを規定する

OpenID Connectとは

OAuth 2.0の仕組みに乗っかって認証の仕組みを規定したもの

OAuth 2.0

OAuth 2.0が決めるもの

- アプリケーションがどうやってアクセストークンを取得するか
- リクエスト・レスポンス（成功・失敗）の仕様

用語説明

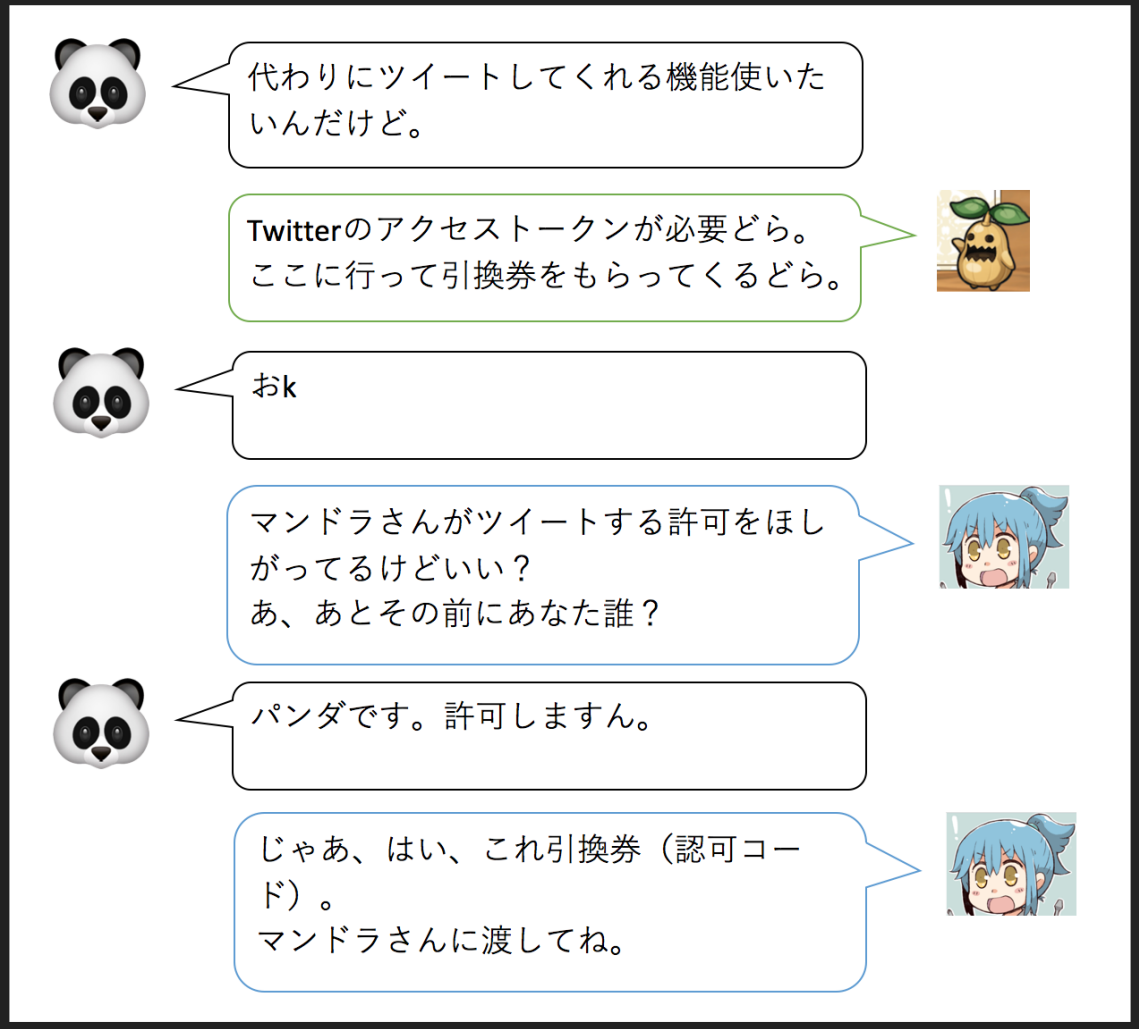
- アクセストークン(Access Token)
  - ユーザーリソースにアクセスするためのトークン
- 認可グラント(Grant)
  - アクセストークンを取得するための方法
  - OAuth 2.0では4つ定義されている
- クレデンシャル(Credential)
  - 認証情報

登場人物

- リソースオーナー
  - ユーザーのこと 🐼
- リソースサーバー
  - ユーザーデータを持っているサーバー 
- クライアント
  - OAuthを使うアプリケーション 
- 認可サーバー
  - ユーザーから認可を取得し、アクセストークンを発行するサーバー 

プロセスフロー

※ 認可コード(AuthorizationCode)Grantの例



引換券（認可コード）もらってきた。

ちょっと待つどら。

引換券どら。アクセストークンをよこすどら。

さっき🐼さんに渡したやつだね。
はい、アクセストークン。

これで🐼の代わりに有る事無い事ツイートできるようになったどら。

🐼の代わりに「レイバンのサングラス、目元の模様にぴったり！」ってツイートするどら。
アクセストークンはこれどら。

正しいアクセストークンだね。
ツイートしたよ。

ふぁ～

注目点

- ユーザー()の認証は認証サーバー()が必要だからやってる
- クライアント()はユーザー()の認証情報を知らない
- リソースオーナー()はアクセストークンを知らない
- リソースサーバー()はユーザーの認証情報やアクセストークンの入手経路は気にしない。アクセストークントークンが正しいかどうかだけが問題

ここで話さなかったこと

- アクセストークンのスコープ(scope)
  - 何を許可するか
- リフレッシュトークンとトークンエンドポイント
  - 毎回アクセストークンを発行するのはコストが高い
- クライアント登録
  - 認証情報以外にリダイレクトURLも登録する
- CSRF対策(state)等セキュリティの細かい話

詳細はWebで！ https://openid-foundation-japan.github.io/rfc6749.ja.html

OpenID Connect

OpenID Connectが決めるもの

OAuth 2.0で行われるユーザーの認証はおまけだった。OpenID Connectではアクセストークンと一緒に認証結果を返したり、アクセストークン取得後にユーザー情報を取得するためのエンドポイントに対するリクエスト・レスポンスも仕様化した。

用語説明

- クレーム(Claim)
  - データ(Entity に関する情報の部分集合)
- ID Token
  - 認証結果に使用されるデータ構造
  - JWT形式
- UserInfo Endpoint
  - アクセストークンを使ってユーザー情報を取得するための受け口

登場人物

- End-User
  - ユーザー
- RP(Relying Party)
  - OAuth 2.0でいうところのクライアント
- OP(OpenID Provider)
  - OpenID ConnectをサポートするOAuth 2.0の認証サーバー

プロセスフロー

※ OpenID Connectには3つのフローがあるが、ここではAuthorization Code Flowの例

認可コード（引換券）の取得までは同じ。

認可コードを使ってアクセストークンを取得する際に、アクセストークンと一緒にID Tokenが返される。

ID TokenにはユーザーのID(sub)等が含まれている。

それだけ。

OpenID Connect薄くない？

ID Token仕様

2. ID Token

TOC

End-Users の認証を可能にするために OpenID Connect が OAuth 2.0 に行う主要な拡張機能は ID トークンデータ構造である.ID トークンは, あるクライアントを使用している際に認可サーバーによる End-User の認証についてのクレームと, 潜在的な他の要求されるクレームを含むセキュリティトークンである.

OpenID Connect で使用されている全ての OAuth 2.0 のフローのために, ID トークン内で以下のクレームが使用される:

iss
: 必須(REQUIRED).レスポンスの発行者のための Issuer 識別子.`iss` の値は, クエリやフラグメントの要素を含まないスキーム, ホスト, 任意のポート番号とパスからなる `https` スキームを使用した大文字と小文字を区別する URL である.

sub
: 必須(REQUIRED).Subject 識別子.例えば, `24400320` や `AItOawmwtWwcT0k51BayewNvutrJUqsvl6qs7A4` のようなクライアントに使われることを意図し, End-User のための発行者内で一意であり決して再割り当てされない識別子である.ASCII で255文字を超えてはならない(MUST NOT).`sub` の値は大文字と小文字を区別する文字列である.

aud
: 必須(REQUIRED).この ID トークンを対象とする Audience(s).audience の値として Relying Party のOAuth 2.0 の `client_id` を含まなければならない(MUST).また他の audiences のための識別子を含んでもよい(MAY).一般的な場合には, `aud` の値は大文字と小文字を区別する文字列の配列である. audience が1つである際の共通の特殊な場合には, `aud` の値は大文字と小文字を区別した文字列でもよい(MAY).

exp
: 必須(REQUIRED).それ以降, ID トークンを処理のために受け入れてはいけない有効期限(MUST NOT).このパラメーターの処理は, 現在の日付/時刻が値の中に記載されている有効期限の日付/時刻以前でなければならない(MUST).実装者は, クロック・キューを考慮するために, 通常は数分間以下の多少の余地を提供してもよい(MAY).その値は, その日付/時刻まで UTC で計測される 1970-01-01T0:0:0Z から秒数を表す JSON 番号である.通常, そして特殊な UTC の日付/時刻に関する詳細は **RFC 3339** [RFC3339] を参照すること.

iat
: 必須(REQUIRED).JWT が発行された時刻.その値は, その日付/時刻まで UTC で計測される 1970-01-01T0:0:0Z から秒数を表す JSON 番号である.

auth_time
: End-User の認証が行われた時刻.その値は, その日付/時刻まで UTC で計測される 1970-01-01T0:0:0Z から秒数を表す JSON 番号である.`max_age` リクエストが生成されたとき, あるいは `auth_time` が不可欠なクレームとして要求されたとき, そのクレームは必須である(REQUIRED).または, それを含めることは任意である(OPTIONAL).(`auth_time` クレームは意味的に OpenID 2.0 **PAPE** [OpenID.PAPE] `auth_time` レスポンスパラメーターに対応する.)

nonce
: 文字列の値は, ID トークンとクライアントセッションを関連づけるため, リプレイアタックの軽減に使用される.その値は, 認証リクエストから ID トークンに変更されずに通過される.ID トークンの中に存在する場合, クライアントはその `nonce` クレームの値が認証リクエストで送信された `nonce` パラメーターの値と同じであるか検証しなければならない(MUST).認証リクエストに存在する場合, 認可サーバーは認証リクエストで送信された nonce の値であるクレーム値を用いて ID トークンの中に `nonce` クレームを含めなければならない(MUST).認可サーバーは使用される `nonce` 値でその他の処理を行うべきではない(SHOULD).`nonce` 値は大文字と小文字を区別する文字列である.

acr
: 任意(OPTIONAL).Authentication Context Class Reference.認証が成功して実行される認証コンテキストクラスを識別する認証コンテキストクラスリファレンスの値を指定する文字列."0" の値は, End-User の認証が **ISO/IEC 29115** [ISO29115] level 1 の要求に合っていないことを示す.例えば, 長期間有効なブラウザー Cookie を使用する認証が "level 0" の使用は適切である例である.level 0 を用いた認証は金銭的な値のリソースへのアクセスを認可するために使用すべきではない(SHOULD NOT).(これは OpenID 2.0 **PAPE** [OpenID.PAPE] `nist_auth_level` 0 に対応する.) 絶対 URI あるいは **RFC 6711** [RFC6711] の予約語は `acr` 値として使用すべきであり(SHOULD), 予約語は登録されているものとは異なる意味で使用してはいけない(MUST NOT).このクレームを使用している Relying Parties はコンテキスト固有であるかもしれない使用されているその値の意味について同意する必要がある.`acr` 値は大文字と小文字を区別する文字列である.

amr
: 任意(OPTIONAL).Authentication Methods References.認証の中で使用される認証のための識別子となる文字列の JSON 配列である.例えば, 値はパスワードと OTP 認証方法の両方が使用されることを示すかもしれない.`amr` クレームで使用される特殊な値の定義は, この仕様の範囲を超える.Relying Parties はコンテキスト固有であるかもしれない使用されているその値の意味について同意する必要がある.`amr` 値は大文字と小文字を区別する文字列である.

azp
: 任意(OPTIONAL).Authorized party.ID トークンを発行された Relying Partyである.これがある場合, この party の OAuth 2.0 の Client ID を含めなければならない(MUST).ID トークンは1つの audience 値が含まれ, その audience が authorized party とは異なるときにこのクレームのみ必要である.Authorized party が単一の audience として同じときそれは含まれているかもしれない.`azp` 値は文字列または URI を含む大文字と小文字を区別する文字列である.

ID トークンは他のクレームを含むかもしれない(MAY). 解明されない使用されるいくつかのクレームは無視しなければならない(MUST).この仕様によって定義される追加のクレームのための **3.1.3.6**, **3.3.2.11**, **5.1** と **7.4** セクションを参照すること.

ID トークンは **JWS** [JWS] によって署名される, **JWS** [JWS] と **JWE** [JWE] のそれぞれによって署名された後に暗号化の両方をされていなければならない(MUST).これにより各 **Section 16.14** に, 認証, 完全性, 否認防止, および任意で機密性を提供する. ID トークンが暗号化されている場合, **[JWT]** で定義されている入れ子 JWT である結果を用いて署名されてから暗号化されなければならなく(MUST), 認可エンドポイントから ID トークンのない戻り値を使用するレスポンスタイプ(認可コードフローを用いているときこのようになる)で, 登録時に明示的に `none` の使用を要求したクライアントでない限り, ID トークンは `alg` 値として `none` を使用してはいけない(MUST NOT).

ID トークンは JWS あるいは JWE の `x5u`, `x5c`, `jku` と `jwk` ヘッダーパラメーターフィールド を使用すべきではない.代わりに, 使用されるキーへの参照は各 **Section 10** に Discovery と Registration parameters を用いて事前に伝えられる.

ユーザー情報の基本定義

5.1. Standard Claims

TOC

この仕様は標準クレームのセットを定義する. それらは **Section 5.3.2** の UserInfo レスポンス内 あるいは **Section 2** の ID トークン内で返却されるように要求できる.

Member	Type	Description
sub	string	Subject - Issuer における End-User の識別子
name	string	End-User の表示用フルネーム. 肩書きや称号 (suffix) 等が含まれることもある. 氏名等の表示順は End-User の locale や選好に従う.
given_name	string	End-User の名 (given name / first name). 複数の given name を持つ文化圏もあることに注意. 全ての given name が含まれる可能性があり, その場合はスペース区切りで表記される.
family_name	string	End-User の姓 (surname / last name). 複数の family name を持つ文化圏や family name を持たない文化圏もあることに注意. 全ての family name が含まれる可能性があり, その場合はスペース区切りで表記される.
middle_name	string	End-User の middle name. 複数の middle name を持つ文化圏もあることに注意. 全ての middle name が含まれる可能性があり, その場合はスペース区切りで表記される. また middle name を使用しない文化圏もある.
nickname	string	End-User のニックネーム. これは `given_name` と同じこともあれば 異なることもある. 例えば, `nickname` が `Mike`, `given_name` が `Michael` として 返されることがある.
preferred_username	string	`janedoe` や `j.doe` といった, End-User の選好する簡略名. これは `@` や `/` や 空白文字といった特殊文字を含むあらゆる valid な JSON 文字列でありうる (MAY). **Section 5.7** にあるように, RP はこの値がユニークであるという 前提で扱ってはならない (MUST NOT).
profile	string	End-User のプロフィールページの URL. この Web ページに掲載されるコンテンツはこの End-User に関するものであろう (SHOULD).
picture	string	End-User のプロフィール画像の URL. この URL は画像ファイル (PNG, JPEG, GIF 画像ファイル等) を参照すること (MUST). またこの画像は End-User が撮影した任意の写真ではなく, End-User に言及する際に 適切な画像とするべきである (SHOULD).
website	string	End-User の Web ページやブログの URL. この Web ページは End-User 自身や End-User が所属する組織が発信する情報を含むべきである (SHOULD).
email	string	End-User の選好する Email アドレス. **RFC 5322** [RFC5322] の addr-spec シンタックスに従うこと. **Section 5.7** にあるように, RP はこの値が ユニークであるという前提で扱ってはならない (MUST NOT).
email_verified	boolean	End-User の Email アドレスが検証済であれば true, さもなければ false. Claim Value が `true` の場合, これは OP が検証を行った 時点において該当 Email アドレスが End-User のコントロール下にあるという確認処理を行ったことを示す. Email アドレスの検証の詳細についてはコンテキストに依存し, 関係者間での トラストフレームワークや契約上の同意事項等などによって異なる.
gender	string	End-User の性別. 本仕様では `female`, `male` を定義する. 定義済の値に適切なものがない場合, その他の値を利用してもよい (MAY).
birthdate	string	End-User の誕生日. **ISO 8601:2004** [ISO8601-2004] `YYYY-MM-DD` 形式で表現される. 生年を `0000` とすることで生年を省略することもできる (MAY). 生年のみを提示する場合は `YYYY` 形式としても良い. プラットフォームの日付関連の関数の実装によって, 生年のみを提供した場合の月日の扱いは 様々であるため, 実装者はこの点を考慮にいれて日付を処理すべきである.
zoneinfo	string	End-User のタイムゾーンを示す zoneinfo **[zoneinfo]** に登録された文字列. 例えば, `Europe/Paris` や `America/Los_Angeles`.
locale	string	End-User の locale. **BCP47** [RFC5646] 言語タグ表現. これは通常 **ISO 639-1 Alpha-2** [ISO639-1] 言語コードを小文字表記, **ISO 3166-1 Alpha-2** [ISO3166-1] 国コードを大文字表記し, ダッシュでつなげたものである. (`en-US`, `fr-CA` 等) 実装によってはダッシュの代わりにアンダースコアを区切り文字に用いる場合もあるため, 互換性の観点からは注意すること. (`en_US` 等) Relying Party はダッシュ区切りに加えてアンダースコア区切りのシンタックスを受け入れてもよい (MAY).
phone_number	string	End-User の選好する電話番号. この Claim のフォーマットとしては **E.164** [E.164] を推奨する (RECOMMENDED). (`+1 (425) 555-1212`, `+56 (2) 687 2400` 等) 電話番号が拡張を含む場合, その拡張は **RFC 3966** [RFC3966] 拡張シンタックスで 表記することを推奨する (RECOMMENDED). (`+1 (604) 555-1234;ext=5678` 等)
phone_number_verified	boolean	End-User の電話番号が検証済であれば ture, さもなければ false. Claim Value が `true` の場合, これは OP が検証を行った時点において該当電話番号が End-User のコントロール下に あるという確認処理を行ったことを示す. 電話番号の検証の詳細についてはコンテキストに依存し, 関係者間でのトラストフレームワークや契約上の同意事項等などによって異なる. また true の場合, `phone_number` Claim は E.164 形式で なければならず (MUST), すべての拡張は RFC 3966 形式でなければならない (MUST).
address	JSON object	End-User の選好する郵送先住所. `address` メンバーは **Section 5.1.1** に定義された メンバーを含む JSON **[RFC4627]** 構造である.
updated_at	number	End-User に関する情報の最終更新日時. この値は UTC 1970-01-01T0:0:0Z から該当時刻までの秒数を示す JSON 数値である.

Table 1: Registered Member Definitions

他にも

- 暗号化
- 署名
- 検証
- その他セキュリティ対策 etc...

薄くはない

冒頭のエラーに戻る

invalid_grant: Invalid JWT: Token must be a short-lived token and in a reasonable timeframe

invalid_grant

OAuth 2.0のアクセストークン取得時のエラーレスポンスに定義されているエラーコード

JWT, timeframe

OpenID ConnectのID Tokenの仕様で

> *実装者は, クロック・キューを考慮するために, 通常は数分間以下の多少の余地を提供してもよい(MAY)*

また、ID Tokenの検証で

iat Claim は現在時刻からはるか昔に発行されたトークンを拒絶するために利用でき, 攻撃を防ぐために nonce が保存される必要がある期間を制限する. 許容できる範囲は Client の仕様である.

って感じであのエラーはOAuth 2.0またはOpenID Connectの現在時刻絡みのエラーかな、ということがわかる。

まとめ

OAuth 2.0

- ユーザーが安全にリソースアクセス許可（認可）を出すための仕組み
- アクセストークンが大事
- 仕組みは安全だけど安易に許可したりアプリケーション自体が信用できなかったりすると意味無い
- （あと実装者が実装サボった場合も危険）

OpenID Connect

- OAuth 2.0前提で認証に関して規定したもの
- アクセストークンを使ったユーザー情報の取得まで決められている
- JWTがガンガン出てくる
- 現在時刻とかにも結構うるさい

参考URL

OAuth 2.0 https://openid-foundation-japan.github.io/rfc6749.ja.html

OpenID Connect https://openid-foundation-japan.github.io/openid-connect-core-1_0.ja.html

Google Cloud Platform Auth ガイド
https://cloud.google.com/docs/authentication

Using OAuth 2.0 to Access Google APIs
https://developers.google.com/identity/protocols/OAu