OAuth2.0

作者：crowds21
原文链接：https://ld246.com/article/1621788146210
来源网站：链滴

OAuth2.0

本文总结一些互联网上已有的简单明了的介绍了该协议的内容,详细理解建议把参考的文章都阅读以下我只是摘取了一些主要的部分,帮助自己理解.

参考:

- OAuth 2.0 的四种方式 - 阮一峰的网络日志 (ruanyifeng.com)
- 详解OAuth 2.0授权协议（Bearer token)
- OAuth 2.0 筆記 (6) Bearer Token 的使用方法 – Yu-Cheng Chuang’ s Blog (yorkxin.org)

简介

一个授权机制(协议),用于给第三方应用授权,来获取用户数据.为了让第三方获取这些数据,会授予第三一个短期有效的令牌(Token),用来代替账号密码. 是 Open Authorization 的简写.

- 与账号密码的不同点
  - 短期有效
  - 可以被撤回
  - 有权限范围,即签发人员可以限定该Token拥有者的权限

具体关于OAuth2.0的内容可以参见==RFC 6749==

- 英文原文: RFC 6749 - The OAuth 2.0 Authorization Framework (ietf.org)

OAuth2.0四种方式

这四种方式分别为:

- 授权码 Authorization Code
- 隐藏式
- 密码式
- 客户端凭证

不管采用哪一种方式,都必须要得到系统的备案,说明身份,然后拿到两个身份识别码

- 客户端ID ClientID
- 客户端密匙 Client Secret

备案是为了方式令牌为滥用,没有备案过的**第三方应用**是不会拿到令牌的.

注意:是第三方应用来获取用户信息

方式一:授权码 Authorization Code

第三方应用申请一个授权码,然后通过该授权码获取令牌

令牌是存储在后端的,所有的与资源服务器的通信都是在后端完成的,这样的前后端分离,可以避免令牌泄露.

```
sequenceDiagram
    autonumber
    a.com->>+b.com: 请求授权码
    b.com-->>-a.com: 返回授权码
    a.com->>+b.com: 请求令牌Token
    b.com-->>-a.com: 返回令牌Token
```

过程一:如果将一个用户在B网站的数据,授权给A网站使用.类似于在a.com点击通过第三方登录,向b.co发送请求.

请求内容包括

```
response_type=code&         //要求返回授权码
client_id=CLIENT_ID&        // 让b知道是谁在请求
redirect_uri=CALLBACK_URL&  //b接受或拒绝请求后的转跳地址
scope=read                  // 要求的数据授权范围
```

过程二:用户转跳后,会要求用户登录,然后询问是否同意给A网站授权,如果用户同意,此时B就会转跳到redirect_url 指定的网址.转跳的同时,会返回一个授权码.

第三步:拿到该授权码后,网站b就会在**后端**向b网址请求令牌(即这些操作都是在后台直接进行的,对用户可见)

```
client_id=CLIENT_ID&    //用于确认身份
client_secret=CLIENT_SECRET&    //用于确认身份
grant_type=authorization_code&    //表明采用的授权方式
code=AUTHORIZATION_CODE&    //上一步拿到的授权码
redirect_uri=CALLBACK_URL    //令牌颁发后的回调网址
```

步骤四:向上一步的redirect_rul发送一个JSON数据,其中的access_token字段就是令牌

```
{
  "access_token":"ACCESS_TOKEN",
  "token_type":"bearer",
  "expires_in":2592000,
  "refresh_token":"REFRESH_TOKEN",
  "scope":"read",
  "uid":100101,
  "info":{...}
}
```

方式二: 隐藏式

针对一些纯前端应用,没有后端.这种方式允许前端直接颁发令牌,没有授权码这个步骤,因此被称为隐藏.

```
sequenceDiagram
    autonumber
    a.com->>+b.com: 请求令牌Token
    b.com-->>-a.com: 返回令牌Token
```

方式三:密码式

如果你高度信任某个应用,可以直接将你的账号密码告诉该应用,该应用就会通过你的账号密码来申请牌Token.

```
sequenceDiagram
    autonumber
    a.com->>+b.com: 申请获取账号密码
    b.com-->>-a.com: 返回账号密码
    a.com->>+b.com: 通过账号密码请求令牌Token
    b.com-->>-a.com: 返回令牌Token
```

方式四:凭证式

适用于没有前端的命令行应用,即在命令行中请求令牌.

这种方式给出的令牌，是针对第三方应用的，而不是针对用户的，即有可能多个用户共享同一个令牌。

令牌的使用

得到Token后,A网站就可以直接向B网站的API发送请求,获取数据了.

此时，每个发到 API 的请求，都必须带有令牌。具体做法是在请求的头信息，加上一个Authorization段，令牌就放在这个字段里面。

更新令牌

B 网站颁发令牌的时候，一次性颁发两个令牌，一个用于获取数据，另一个用于获取新的令牌（refres token 字段）。令牌到期前，用户使用 refresh token 发一个请求，去更新令牌。

```
grant_type=refresh_token&    //表示要求更新令牌
client_id=CLIENT_ID&         //确认身份
client_secret=CLIENT_SECRET&    //确认身份
refresh_token=REFRESH_TOKEN        //之前颁发的refresh_token
```

AccessToken的具体用法

OAuth 2.0通过补充规范 RFC 6750 示范了一种access token的具体用法，符合这种具体用法的access token统称为Bearer token。注意：**Bearer token不是一种token值的格式，而是一种规范的用法**，Auth 2.0没有规定token值的内容、格式.

传递给服务器的方式

OAuth 2.0在补充规范RFC 6750 中定义了三种传递access token的方式。

1. 放在请求头中传递：在请求头 Authorization字段中使用Bearer这一关键字传递。资源服务器必支持这种传递方式。

```
GET https://api.amazon.com/user/profile
Authorization: Bearer 2YotnFZFEjr1zCsicMWpAA
```

2. 放在表单请求体中传递：资源服务器自行选择是否支持这种传递方式。

```
POST https://api.amazon.com/user/profile
Content-Type: application/x-www-form-urlencoded

access_token=2YotnFZFEjr1zCsicMWpAA
```

3. 放在URI的查询参数中传递（不建议）：RFC 6750 不建议采用这种方式，如果实在无法使用前两方式，再来考虑这种。资源服务器自行选择是否支持这种传递方式。

accessToken的值

OAuth 2.0没有规定access token值的内容、格式，**只要求access token对客户机应用不透明。**实操常见的格式分为两类：

1. 读库令牌：access token值是一个随机生成的标识符，在数据库中存储有效期，谁颁发的，颁发给的，权限范围等元数据信息。校验时需先读库，优点是当授权服务器想主动撤销已签发的access toke时非常方便。

2. 自包含令牌**（例如JWT）**：access token值是把上述元数据信息编码、签名、加密后得到的值优点是校验时无需读库，缺点是当授权服务器想主动撤销已签发的access token时会比较棘手。

JWT

JSON WEB TOKEN是目前比较流行的自包含令牌格式.

- 基于json的通用性，可以跨语言支持。
- 简单紧凑，字节占用小，能够放在 HTTP 报头或URI的查询参数中进行传输。
- 自包含并提供防篡改机制。

相关的标准文件：

【RFC7519】JSON Web Token (JWT)

【RFC7515】JSON Web Signature (JWS)

【RFC7516】JSON Web Encryption (JWE)

【RFC7517】JSON Web Key (JWK)

【RFC7518】JSON Web Algorithms (JWA)

服务器校验AccessToken

【RFC 7662】OAuth 2.0 Token Introspection

OAuth 2.0没有规定资源服务器如何校验access token，只是说资源服务器与授权服务器之间自己协。实操中一般采用以下方法校验：

对于小型Web应用：资源服务器通常与授权服务器同为一体，自然能够通过读库来校验。

对于大型Web应用：授权服务器和资源服务器通常是独立部署的，有三种方式校验：

1. 使用读库令牌作为access token，在数据库存储层面做共享，使得资源服务器能够通过读库来校验。

2. 使用读库令牌作为access token，由授权服务器提供一个令牌校验接口，资源服务器请求该接口来验。OAuth2.0在补充规范RFC 7662 中定义了令牌校验接口（Introspection Endpoint）的相关标准。

3. 使用自包含令牌作为access token，资源服务器和授权服务器双方约定好自包含令牌的结构、签名钥、加密方法，资源服务器按照约定规则自行校验。

撤销Token

补充规范标准文件：【RFC 7009】OAuth 2.0 Token Revocation

OAuth 2.0在补充规范RFC 7009中定义了一个由授权服务器提供的撤销接口（Revocation Endpoint）供客户机应用申请撤销access_token或refresh_token。

当用户在客户机应用退出登录、更换账号、注销账号，或卸载了客户机应用时，客户机应用需要通知权服务器自己不再需要该用户的令牌，授权服务器将清除与该令牌相关的授权信息。 这样可以防止被弃令牌的滥用，并改善用户体验，因为失效的授权将不再出现在授权服务器展示给用户的已授权客户应用列表中。

其他

回调地址

即下一步要转跳或者发送请求的地址

一个向Github发送请求的实例

GitHub OAuth 第三方登录示例教程 - 阮一峰的网络日志 (ruanyifeng.com)