

Oauth2认证

Oauth2简介

简介

第三方认证技术方案最主要是解决认证协议的通用标准问题，因为要实现跨系统认证，各系统之间要遵循一定的接口协议。

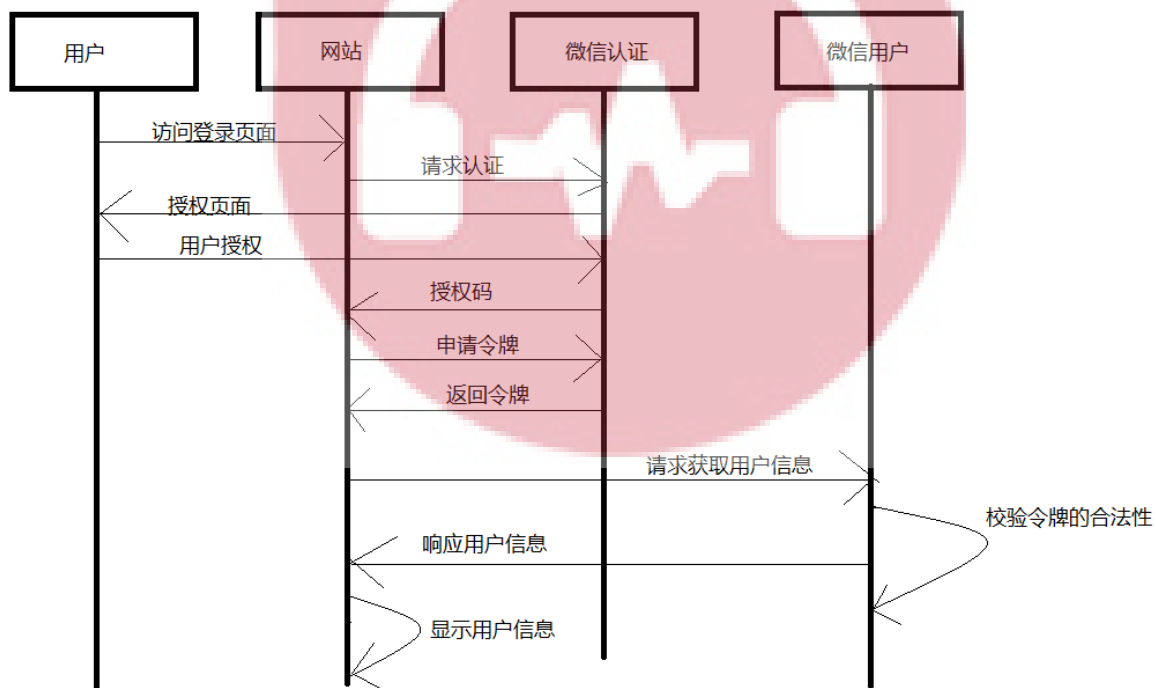
OAuth协议为用户资源的授权提供了一个安全的、开放而又简易的标准。同时，任何第三方都可以使用OAuth认证服务，任何服务提供商都可以实现自身的OAuth认证服务，因而OAuth是开放的。业界提供了OAuth的多种实现如PHP、JavaScript、Java、Ruby等各种语言开发包，大大节约了程序员的时间，因而OAuth是简易的。互联网很多服务如Open API，很多大公司如Google、Yahoo、Microsoft等都提供了OAuth认证服务，这些都足以说明OAuth标准逐渐成为开放资源授权的标准。

OAuth协议目前发展到2.0版本，1.0版本过于复杂，2.0版本已得到广泛应用。

参考：<https://baike.baidu.com/item/oauth/7153134?fr=aladdin>


OAuth 协议：<https://tools.ietf.org/html/rfc6749>

下边分析一个OAuth2认证的例子，网站使用微信认证的过程：



1. 用户进入网站的登录页面，点击微信的图标以微信账号登录系统，用户是自己在微信里信息的资源拥有者。

扫码登录 更快更安全



帐号登录

短信登录

输入手机号，短信登录更快捷

验证码


3321

动态密码


获取动态密码

马上登录


合作账号一键登录




微信



QQ



新浪微博



百度

点击“微信”出现一个二维码，此时用户扫描二维码，开始给网站授权。





微信登录



2. 资源拥有者同意给客户端授权

资源拥有者扫描二维码表示资源拥有者同意给客户端授权，微信会对资源拥有者的身份进行验证，验证通过后，微信会询问用户是否给授权网站访问自己的微信数据，用户点击“确认登录”表示同意授权，微信认证服务器会颁发一个授权码，并重定向到网站。

3. 客户端获取到授权码，请求认证服务器申请令牌

此过程用户看不到，客户端应用程序请求认证服务器，请求携带授权码。

4. 认证服务器向客户端响应令牌

认证服务器验证了客户端请求的授权码，如果合法则给客户端颁发令牌，令牌是客户端访问资源的通行证。此交互过程用户看不到，当客户端拿到令牌后，用户在网站看到已经登录成功。



5. 客户端请求资源服务器的资源

客户端携带令牌访问资源服务器的资源。网站携带令牌请求访问微信服务器获取用户的基本信息。

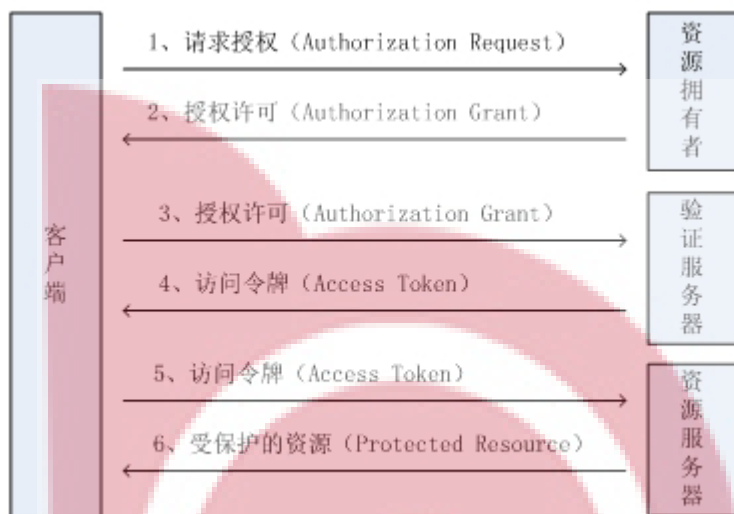
6. 资源服务器返回受保护资源

资源服务器校验令牌的合法性，如果合法则向用户响应资源信息内容。

注意：资源服务器和认证服务器可以是一个服务也可以分开的服务，如果是分开的服务资源服务器通常要请求认证服务器来校验令牌的合法性。

Oauth2.0认证流程如下：

引自Oauth2.0协议rfc6749 <https://tools.ietf.org/html/rfc6749>



角色

客户端

本身不存储资源，需要通过资源拥有者的授权去请求资源服务器的资源，比如：Android客户端、Web客户端（浏览器端）、微信客户端等。

资源拥有者

通常为用户，也可以是应用程序，即该资源的拥有者。

授权服务器（也称认证服务器）

用来对资源拥有的身份进行认证、对访问资源进行授权。客户端要想访问资源需要通过认证服务器由资源拥有者授权后方可访问。

资源服务器

存储资源的服务器，比如，网站用户管理服务器存储了网站用户信息，网站相册服务器存储了用户的相册信息，微信的资源服务存储了微信的用户信息等。客户端最终访问资源服务器获取资源信息。

常用术语

- 客户凭证(client Credentials)：客户端的clientId和密码用于认证客户
- 令牌(tokens)：授权服务器在接收到客户请求后，颁发的访问令牌
- 作用域(scopes)：客户请求访问令牌时，由资源拥有者额外指定的细分权限(permission)

令牌类型

- **授权码**：仅用于授权码授权类型，用于交换获取访问令牌和刷新令牌
- **访问令牌**：用于代表一个用户或服务直接去访问受保护的资源
- **刷新令牌**：用于去授权服务器获取一个刷新访问令牌
- **BearerToken**：不管谁拿到Token都可以访问资源，类似现金
- **Proof of Possession(PoP) Token**：可以校验client是否对Token有明确的拥有权

特点

优点：

更安全，客户端不接触用户密码，服务器端更易集中保护

广泛传播并被持续采用

短寿命和封装的token

资源服务器和授权服务器解耦

集中式授权，简化客户端

HTTP/JSON友好，易于请求和传递token

考虑多种客户端架构场景

客户可以具有不同的信任级别

缺点：

协议框架太宽泛，造成各种实现的兼容性和互操作性差

不是一个认证协议，本身并不能告诉你任何用户信息。