第32卷第6期 佛山科学技术学院学报(自然科学版) Vol.32 No.6
2014年11月 Journal of Foshan University(Natural Sciences Edition) Nov. 2014

文章编号:1008-0171(2014)06-0051-05

解析 OAuth 2.0 流程及认证接口设计

吴栋淦

(福建信息职业技术学院 计算机工程系,福建 福州 350003)

摘要:OAuth 2.0 框架能够在无需向第三方应用泄露用户机密的情况下允许第三方应用访问受保护资源。与旧版本相比,OAuth 2.0 提供了更简洁和更安全的环境。研究了 OAuth 2.0 的工作流程,分析了认证客户端的设计方案并给出一个客户端应用实例。

关键词:OAuth 2.0;授权;访问令牌

中图分类号:TP393.08 **文献标志码**:A

近几年来,国内外许多 WEB 2.0 应用受到广泛的关注,诸如 Flickr、Twitter、Facebook、新浪微博、人人网等应用无不热火朝天。这些应用基本上都使用了开放平台来进行构建,是将常用的服务封装到 API 接口中,并开放 API 接口,第三方开发人员可以使用这些 API 接口来进行复杂的数据交换。

开放平台的认证技术是 API 接口需要重点考虑的问题,认证技术需要保障用户能简单有效地使用资源,并确保用户凭证(比如用户名、口令)不被泄露。目前主流的开放平台大部分使用 OAuth 授权技术,OAuth 作为一个开放标准,在无需向第三方泄露用户名和密码的情况下允许第三方访问私密的数据[1]。

1 OAuth 授权技术

1.1 OAuth 协议

OAuth 协议当前使用 1.0A 和 2.0 两个版本。2010 年,互联网工程任务组 IETF 在 RFC 5849[2]中定义了 OAuth Core 1.0A 版本,Twitter、Flickr、新浪微博等平台都先后使用了 OAuth 1.0A 认证协议。自 2010 年 4 月 IETF 公布 OAuth 2.0 草案开始,各大平台先后宣布对尚未定稿的 OAuth 2.0 进行支持,人人网是国内首先使用 OAuth 2.0 的平台[3],此外,新浪微博、开心网也相继使用了 OAuth 2.0。直到 2012 年 10 月,IETF 在 RFC 6749 中对 OAuth 2.0 版本定稿,OAuth 2.0 的使用已是大势所趋。OAuth 1.0A 和 OAuth 2.0 的主要区别在于:OAuth 2.0 对认证流程进行了简化。

1.2 OAuth 术语

资源拥有者(Resource Owner):能够批准对受保护资源进行访问的实体。

资源服务器(Resource Server):通过使用访问令牌,能够接收受保护资源访问请求并对该请求进行应答的服务器。

客户端(Client):代表资源拥有者对受保护资源提出访问请求的应用。

授权服务器(Authorization Server):在成功与资源拥有者进行认证并获得授权后,向客户端颁发访问令牌的服务器。

授权许可(Authorization Grant):资源拥有者对客户端授权的凭证,客户端使用授权许可来交换访问

收稿日期:2014-05-22

作者简介:吴栋淦(1979-),男,福建浦城人,福建信息职业技术学院讲师。

令牌,RFC 6749 标准定义了四种类型的授权许可及扩展类型的授权许可。

访问令牌(Access Token):由授权服务器颁发的惟一标识符,是访问受保护资源的凭证。

刷新令牌(Refresh Token):授权服务器颁发给客户端的一种凭证,当访问令牌非法或过期时使用刷新令牌来获取新的访问令牌。

服务接口(Endpoint):站点向用户提供服务接口用于完成数据交换和业务服务[4]。

2 OAuth 2.0 协议流程

在 OAuth 2.0 中,客户端通过访问令牌来访问受保护资源,其认证流程如图 1 所示,总共有如下三个阶段。

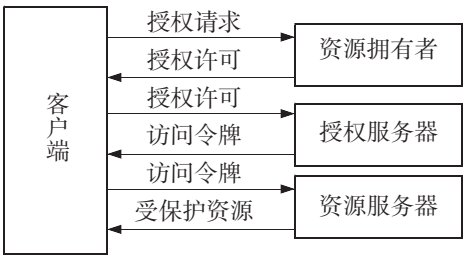

图 1 OAuth 2.0 协议流程

(1)客户端向资源拥有者申请获得授权,申请可以直接发往资源拥有者,也可以通过授权服务器为中介(步骤 1),资源拥有者许可授权,并向客户端发送授权许可(步骤 2),授权许可代表着资源拥有者授权的凭证,可以是四种标准许可类型(或扩展许可类型)中的一种;

(2)客户端通过与授权服务器进行认证,并出示授权许可来申请访问令牌(步骤 3),授权服务器与客户端完成认证并确保授权许可合法性,向客户端颁发访问令牌(步骤 4);

(3)客户端向资源服务器出示访问令牌来申请对受保护资源的访问(步骤 5),资源服务器验证访问令牌的合法性,向客户端提供服务(步骤 6)。

2.1 获得授权许可

此阶段的目的是为了获得授权许可,标准定义的四种基本授权许可类型分别为授权码、隐式、资源拥有者口令凭证和客户端凭证,此外用户可以自定义扩展的授权许可类型。授权过程中需要利用两个服务接口:授权接口和令牌接口,客户端使用授权接口通过用户代理重定向从资源拥有者获得授权,另外客户端使用令牌接口来获得访问令牌。

为了获得授权许可,客户端需要在 HTTP 请求中加入所需的参数并将请求发往授权接口,这些参数包括 response_type、client_id、redirect_uri、scope 和 state。client_id 是一个独一无二的客户端标识符,在客户端注册时颁发客户端标识符,同时还颁发客户端机密(client_secret),客户端标识符不能单独用于客户端认证。redirect_uri 是重定向接口,客户端注册时需要设置该接口,授权服务器通过使用重定向接口将资源服务器用户代理回送到客户端。scope 是访问请求的作用域,如果客户端忽略该值,授权服务器必须使用预定义的默认作用域。state 被客户端用于维护请求状态。

当授权许可类型是授权码时,response_type 参数的值必须为 code。当授权码类型是隐式时,response_type 的值为 token。当授权码类型为资源拥有者口令凭证时,response_type 的值为 password,客户端必须向服务器提交客户标志、客户端标识符和客户端机密。当授权码类型为客户端凭证时,客户端只需要将值为 client_credentials 的 response_type 参数发送。

当客户端使用授权码许可类型且资源拥有者批准授权请求,需要在给客户端的应答中颁发授权码,应答中可能包含的参数有 code 和 state。code 是授权服务器生成的授权码,为了降低安全风险,授权码必须在颁发后尽快失效,规范中推荐授权码最大生存时间为 10 min。在客户端授权请求时如指定了 state,则授权服务器的应答中 state 必须与从客户端接收到的值保持一致。如果客户端使用其他三种客户端凭证类型将在此步骤直接获得访问令牌。

OAuth 2.0 相比 OAuth 1.0 做了较大简化,比如在 OAuth 1.0 中需要向服务接口发送由 HTTP 请求 URL 和其他参数计算而来的签名值,而在 OAuth 2.0 中不再计算签名,此外,OAuth 2.0 也取消了 oauth_timestamp 和 oauth_nonce 两个参数。同时,OAuth 2.0 对认证过程中的数据传输保密性提出了更高

的要求,强制使用 TLS 来保障数据安全。

2.2 获得访问令牌

授权码类型客户端获得授权许可后,需要向授权服务器发送 HTTP 请求以获得访问令牌。HTTP 请求包括以下参数:grant_type、code、redirect_uri 和 client_id。当平台使用授权码许可类型时,grant_type 值必须为“authorization_code”。code 参数值是在授权许可步骤中从授权接口获得的授权码。redirect_uri 和 client_id 与前一步骤相同。

授权服务器收到 HTTP 请求后,需要对客户端进行认证,并验证授权码合法性。如果通过授权码合法性验证,授权服务器需要向客户端发送应答,颁发访问令牌 access_token 和可选的刷新令牌 refresh_token。此外,应答中可能还包括 token_type 和 expires_in 参数,其中 token_type 参数是访问令牌类型,expires_in 参数来指定访问令牌的有效期。

2.3 访问受保护资源

客户端通过向资源服务器出示访问令牌来访问受保护资源,资源服务器需检查访问令牌,确保访问令牌合法。如果访问令牌合法则正常受理访问请求,如果令牌过期,则要求客户端重新获取访问令牌。如果在此之前授权服务器已经向客户端颁发过刷新令牌,则使用刷新令牌来获取新的访问令牌。

3 OAuth 2.0 客户端认证接口实例

本文以某社交平台授权机制为例,分析 OAuth 2.0 客户端注册和认证的流程,并以 Cocoa Touch 为基础实现了认证客户端。

3.1 客户端注册

第三方开发人员在设计新的客户端之前需要在官方网站上注册应用,注册信息包括应用名称、应用平台、应用介绍等,注册成功之后将获得 App Key 和 App Secret,这两个字符串分别对应 client_id 和 client_secret 参数。

3.2 服务接口

开放平台提供了若干服务接口(API),与 OAuth 2.0 认证相关的服务接口有五个,其中两个 API 接口将在下文中使用到,分别是请求授权和获取授权。

请求授权用于图 1 流程中第 1、2 步骤,地址 https://api.xxx.com/oauth2/authorize。获取授权用于图 1 流程中第 3 和第 4 步骤,地址 https://api.xxx.com/oauth2/access_token。

3.3 客户端实现及安全考虑

为了便于认证客户端的实现,可以定义一个 OAuth2Utility 类,下面是类中几个主要的方法。

(void)authorizationRequest;// 申请获得授权

(void)authorizationResponse:(NSString *)authorizationCode clientState:(NSString *)state;// 客户端接收到授权许可

(void)accessTokenRequest:(NSString *)authorizationCode grantType:(NSString *)type;// 使用授权许可申请访问令牌

(void)accessTokenResponse:(ASIHTTPRequest *)request;// 客户端接收到访问令牌

在移动客户端 API 中,申请访问授权需要提交的参数包括 client_id、response_type 和 redirect_uri,如需要支持移动客户端 html5 特性,需提交参数 display,并将该值设置为“mobile”。所有参数需要使用 x-www-form-urlencoded 格式进行编码,并通过 HTTP GET/POST 方式发送。比如使用授权码类型的客户端发送的 HTTP 请求可能是 https://api.xxx.com/oauth2/authorize ?client_id=s6BhdRkqt3&response_type=code&redirect_uri=http%3A%2F%2Fclient&display=mobile。RFC6749 规范定义的四种标准授权许可类型中,授权码许可被广泛支持,而在隐式许可类型中,访问令牌作为重定向 URI 的一个部分直接返回给客户端,不颁发刷新令牌。资源拥有者口令凭证许可类型和客户端凭证许可类型由于存在巨大的

安全风险,一般只在特别的场合下使用。

开放平台的请求授权服务接口将授权码通过用户代理发回给客户端,在本例中服务接口发回客户端的 HTTP 应答可以是 http://localhost/?code=e8dc2e18d6382cf212598e60b6fe7c5e,URL 中的 code 值就是授权码。不少服务接口并没有完全遵循规范的要求进行设计,比如本文的例子,容易造成很多安全隐患。Homakov[5]利用跨站请求伪造(CSRF)攻击来非法访问第三方资源,攻击者可以利用这个被挟持的账号继续挟持其他网站的账号[6]。此外,攻击者还可以使用猜测授权码、恶意客户端授权、授权码钓鱼、用户会话模仿等方式来获得授权码。搜狐微博 OAuth 2.0 授权曾经由于设计不当,被发现在获取授权码的时候没有返回 state 参数而造成安全隐患。为了规避安全风险,建议服务接口设计时应采纳 state 参数,并在客户端注册期间使用完整的重定向 URI[7]。此外,为了减少授权码重放攻击带来的危害,应该尽可能缩短授权码的有效期。

客户端获得授权码之后就可以向获取授权服务接口要求交换访问令牌,交换访问令牌需要提交的参数包括 client_id、client_secret、grant_type, 如果 grant_type 类型为授权码时, 还必须提交 code 和 redirect_uri 参数。所有参数经过编码之后,通过 HTTP POST 方法发送到获取授权服务接口。HTTP 发送的数据可以是 https://api.xxx.com/oauth2/access_token?code= e8dc2e18d6382cf212598e60b6fe7c5e&redirect_uri=http% 3A% 2F% 2Fclient&grant_type=authorization_code&client_secret=1fbc37441170986&client_id=s6BhdRkqt3。

获取授权服务接口对客户端提交的参数进行验证,如验证通过则向客户端发送应答,应答中包括 access_token、expires_in、remind_in 和 uid。access_token 是颁发的访问令牌。expires_in 是令牌有效期,令牌有效期取决于令牌泄漏的风险,由许多不同的因素决定。该平台为几种客户端分别设置了不同的令牌有效期,比如测试客户端的有效期为 1 d,普通客户端为 7 d。remind_in 即将废弃,现已使用 expires_in 代替 remind_in。uid 是当前授权用户的编号。服务接口应答数据采用 JSON 编码,JSON 是一种基于 JavaScript 轻量级的数据交换格式。客户端需要使用 JSON 的开发库来帮助解读 JSON 格式数据。

客户端获得访问令牌之后就可以使用令牌访问资源, 比如读取公共信息或者发信息等。以读取公共信息为例,客户端向公共信息服务接口提交访问令牌就能获得最新的 200 条公共信息, 客户端可以通过提交参数 count 来设置单页返回记录的条数。客户端通过 HTTP GET 提交数据,返回的数据使用 JSON 格式编码。

图 2 客户端运行效果截图

3.4 客户端测试

本例中客户端应用平台为 iOS, 客户端在 Mac OS X 10.8.3 系统以及 Xcode 4.6.2 开发环境下测试通过。图 2 是客户端运行的效果截图。

4 结语

OAuth 授权机制在许多第三方应用认证中被广泛运用,尤其是 OAuth 2.0 在 OAuth 1.0 基础上简化了业务流程,降低了安全风险。目前很多开放平台已经转向 OAuth 2.0,并逐渐放弃对 OAuth 1.0 的支持。OAuth 协议将为多平台之间的数据交互和认证流程简化提供坚实的保障。

参考文献:

[1] EDWARD D H. The OAuth 2.0 Authorization Framework[EB/OL]. [2012-10-03]. http: //tools. ietf. org/html/rfc6749.

[2] LAHAV H. The OAuth 1.0 Protocol[EB/OL]. [2010-04-01]. http: //tools. ietf. org/html/rfc5849.

[3] 刘镝, 张智江, 张尼. 基于国内开放平台的 OAuth 认证框架研究[J]. 信息通信技术, 2011(6): 43-46.

[4] 张卫全, 胡志远. 浅析作用于 Web2.0 安全防范的 OpenID 和 OAuth 机制[J]. 通信管理与技术, 2011(2): 15-18.

[5] HOMAKOV E. The Most Common OAuth2 Vulnerability[EB/OL]. [2012-07-03]. http: //homakov. blogspot. com/2012/07/saferweb-most-common-oauth2. html.
[6] 张天琪. OAuth 协议安全性研究[J]. 信息网络安全, 2013(3): 68-70.
[7] EDWARD T L. OAuth 2.0 Threat Model and Security Considerations[EB/OL].[2013-01-20]. http: //tools. ietf.org/html/rfc6819.

【责任编辑:王桂珍　foshanwgzh@163.com】

Analyze the OAuth 2.0 process and the design of authenticating interface

WU Dong-gan

(Department of Computer Engineering, Fujian Polytechnic of Information Technology, Fuzhou 350003, China)

Abstract: The OAuth 2.0 framework enables third-party application to access protected resource without exposing the user's credentials to third-party application. The OAuth 2.0 framework provides more simpler and safer environment than the obsoleted version. This paper investigates the procedure of OAuth 2.0, analyzes the design proposal of authentication client and offers the client application example.

Key words: OAuth 2.0; authorization; access token

(上接第 14 页)

The edge-magic total labeling and the super edge-magic total labeling of S^*

DING Sheng[1,2], LIU Jia-bao[3], PEI Li-dan[1]

(1. School of Mathematics Science, Anhui University, Hefei 230601, China;
2. School of Anhui University, Hefei 230039, China;
3. Department of Public Teaching, Anhui Xinhua University, Hefei 230088, China)

Abstract: This paper studies $\forall n \in N^*$ of graphs S^* is the edge-magic total labeling and the super edge-magic total labeling, obtaining two different edge-magic total labeling algorithms A and B, $\forall n \in N^*$ graphs S^* is the super edge-magic total constant C_1=5n+6 and the super edge-magic total constant C_2=7n+6, graphs S^* has n+1 vertices Star graphs $S(u)$and n+1 vertices star graphs $S(v)$, which proves graphs S^* is not only the edge-magic graph,but also the super edge-magic total graphs conclusions.

Key words: star graphs; the super edge-magic total labeling; the super edge-magic total graphs