Oauth2.0 授权框架

概要

OAuth2.0 授权框架可以允许一个第三方的应用程序得到一个有限的访问一个HTTP服务的权限要么代表资源所有者通过编排一个批准互动资源所有者和HTTP服务,或通过允许第三方应用程序来获得代表其自身。这篇说明书替代和淘汰在RFC5849上所描述的OAuth1.0协议.

这份备忘录的状态

这是一份互联网标准跟踪文档. 这份文档是互联网工程任务(IETF)的驱使下的产品.它代表了IETF社区的意志.他接受了公众的审查并且已经被互联网工程指导组(IESG)认可出版.如果想要更加深入的了解的话可以参考文档RFC5741互联网标准的第二节.

版权提醒

1.1. 规则

OAuth定义了四个规则:

资源拥有者 一个实体能够授权访问受保护的资源, 当这个资源的所有者是一个人的时候, 他被成为终端用户.

资源服务器 一个服务器管理着受保护的资源,能够接受和响应对于这些受保护资源的使用访问令牌的请求.

客户端 一个应用程序在资源的所有者和他的授权代表下制造受保护的资源.

授权服务器 一个服务器发放访问令牌给客户端在成功验证资源的拥有者并且获取授权.

在授权服务器和资源服务器的互动已经超出了本规范的范围. 授权服务器可能和资源服务器在同一台服务器上也可能是两个独立的主体.一个单独的授权服务器可能发出的访问令牌被多个资源服务器所接受.

1.2. 协议流

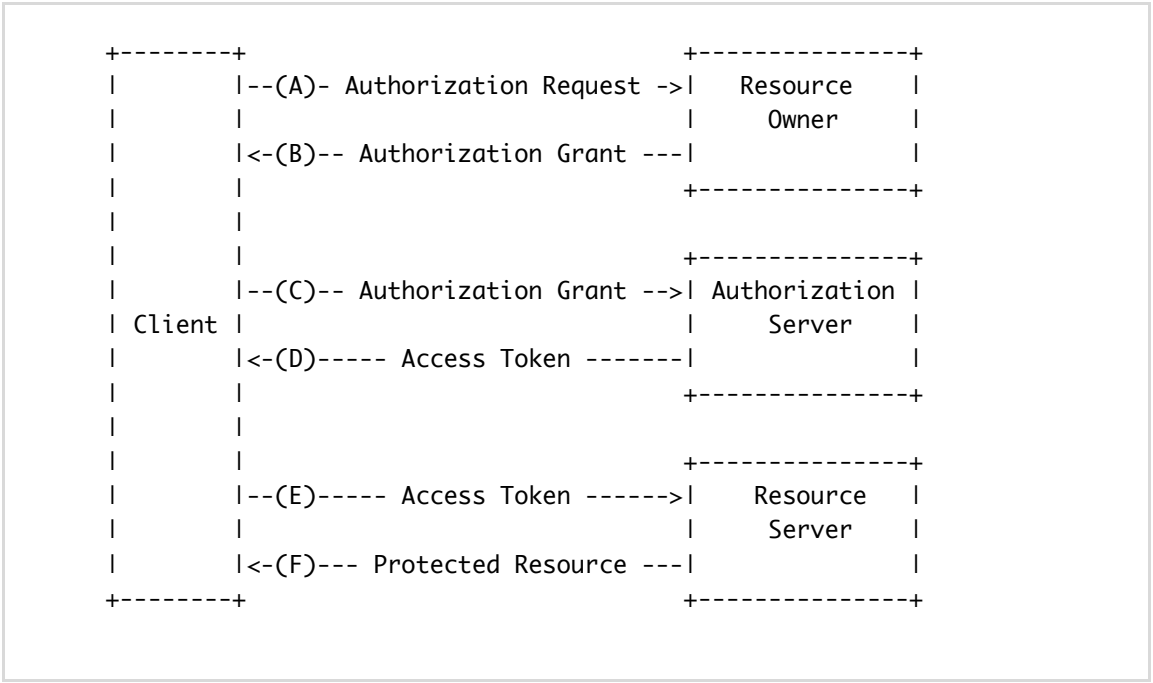

图1: 协议流概要

OAuth2.0概要协议流已经在图1中给出, 描述了四个规则之间的相互作用并且包含了下面几步:

(A) 客户端向资源拥有者请求了授权. 授权请求可以直接被送到资源的拥有者哪里(如上所示), 或者更好的间接通过授权服务器作为中间人.

(B) 客户端接受一个代表了资源拥有者的权限的凭证的授权批准, 可以使用本规范所列举出的四个授权类型的其中一个也可以扩展一个授权类型. 授权类型依靠客户端发出的授权请求所使用的方法和授权服务器所支持的类型.

(C) 客户端通过授权服务器的认证授权请求代表了其权限的访问令牌.

(D) 授权服务器鉴定客户端的真假, 验证授权, 如果验证通过, 颁发一个访问令牌.

(E) 客户端向资源服务器请求受保护的资源通过提供提供访问令牌通过验证.

(F) 资源服务器验证访问令牌, 如果是有效的, 提供服务.

客户端从资源拥有者获取授权认证的首选方案(在步骤(A)和步骤(B)中所指的)是通过授权服务器作为中间人认证, 这个方案在第4.1节中图3中有具体的描述.

1.3. 授权认证(Authorization Grant)

一个授权认证是一个代表资源拥有者的权限(访问他的受保护的资源)的凭证, 被客户端使用获取访问令牌. 这份规范定义了四个授权类型 -- 授权代码(authorization code), 瘾式授权(implicit), 资源拥有者密码证书(resource owner password credentials)和客户端凭证(client credentials) -- 当然也可以是一个可扩展机制下的自定义附加类型.

1.3.1. 授权代码(authorization code)

授权代码是通过授权服务器作为客户端和资源服务器的中间者获得的. 代替了直接向资源所有者请求授权, 客户端通过授权服务器指向资源拥有者(通过他自身的定义在[RFC2616]的用户代理), 依次定向资源拥有者得到授权码回到客户端.

在定向资源拥有者带到授权码回到客户端之前, 授权服务器鉴定资源拥有者并且获得授权. 因为资源拥有者只通过授权服务器授权, 资源拥有者的凭证永远不会共享到客户端.

授权代码提供一个一些小的重要的安全益处, 比如说有能力鉴定客户端, 同样, 认证令牌的传递直接到客户端没有通过资源拥有者的用户代理所以不会潜在的暴露给他人, 包括资源的拥有者.

1.3.2. 隐式授权(implicit)

隐性授权是一种简单的授权码流对在浏览器中执行的脚本代码比如说JavaScript的客户端的优化.在隐式授权流中, 代替颁发给客户端一个授权码, 客户端直接被分配了一个认证令牌(作为资源拥有者的授权的结果). 这种认证类型是隐性的, 因为没有中间的证书(比如说认证码)被分配(后来更加习惯获取一个访问令牌).

1.3.3. 资源拥有者密码证书(resource owner password credentials)

资源拥有者密码证书(i.e., 用户名和密码)可以被直接作为授权认证来使用获取认证令牌. 这种方式只能在资源拥有着和客户端高度信任过的情况下使用(e.g.,客户端是设备操作系统的一部分或者有高度权限的客户端), 或者当其他的授权认证类型不可用(比如说授权代码).

尽管这种授权类型需要直接导向客户端访问资源拥有着的证书,资源拥有着证书被用作一个单一的请求来交换认证令牌.这种授权方式可以消除客户端保存资源拥有着的证书作为以后使用一个长期存活的认证令牌或者刷新证书来交换证书的这种需求.

1.3.4. 客户端凭证(client credentials)

当授权范围仅限于客户端控制下的受保护资源,或是先前使用授权服务器处理过的受保护资源时，应用凭证或其他形式的客户端授权可以用作授权认证。当客户端代表自己的时候（客户端为资源所有者自身），或请求访问基于先前由授权服务器处理过的授权的受保护资源是应用凭证用作认证授权的典型情况。

1.4. 访问令牌(Access Token)

```
Access tokens are credentials used to access protected resources.  An access
token is a string representing an authorization issued to the client.  The
string is usually opaque to the client.  Tokens represent specific scopes
and durations of access, granted by the resource owner, and enforced by the
resource server and authorization server.
```

访问令牌一个用于访问受保护资源的凭证, 是一条代表授权服务器发送给客户端的字符串, 该字符串往往对于客户端是透明的. 令牌由资源拥有着授权, 由资源服务器和授权服务器执行, 指定了访问的范围和时间.

```
The token may denote an identifier used to retrieve the authorization
information or may self-contain the authorization information in a
verifiable manner (i.e., a token string consisting of some data and a
signature).  Additional authentication credentials, which are beyond the
scope of this specification, may be required in order for the client to use
a token.
```

该令牌可以用作获取授权信息的标示符, 也可以以某种方式自身包含授权信息(例如, 一个令牌字符串由一系列的数据和签名构成). 为了客户端使用令牌而用到的附加的认证已经超出了这篇规范.

```
The access token provides an abstraction layer, replacing different
authorization constructs (e.g., username and password) with a single token
understood by the resource server.  This abstraction enables issuing access
tokens more restrictive than the authorization grant used to obtain them, as
well as removing the resource server's need to understand a wide range of
authentication methods.
```

访问令牌提供了一个通过资源服务器认可的唯一 token 去代替不同的授权结构抽象层（如用户名和密码）。这种抽象的方法使发出访问 token 比用于获得令牌的授权认可有更大的限制，同时也不需要资源服务器理解不同的验证方法。

```
Access tokens can have different formats, structures, and methods of
utilization (e.g., cryptographic properties) based on the resource server
security requirements.  Access token attributes and the methods used to
access protected resources are beyond the scope of this specification and
are defined by companion specifications such as [RFC6750].
```

根据资源服务器的安全需要，访问 Token 可以有不同的格式、结构、和工具方法（比如加密属性）。

1.5. 可刷新令牌

```
Refresh tokens are credentials used to obtain access tokens.  Refresh tokens
are issued to the client by the authorization server and are used to obtain
a new access token when the current access token becomes invalid or expires,
or to obtain additional access tokens with identical or narrower scope
(access tokens may have a shorter lifetime and fewer permissions than
authorized by the resource owner).  Issuing a refresh token is optional at
the discretion of the authorization server.  If the authorization server
issues a refresh token, it is included when issuing an access token (i.e.,
step (D) in Figure 1).
```

可刷新Token是用于获得访问Token的凭证。可刷新Token由授权服务器发放给客户端，用于当前访问Token无效或过期时，可刷新Token可以用于获得一个新的Token令牌或得到一个相同使用范围（或更小）的额外访问Token。（访问Token可能比资源拥有者的授权，有更少的生存时间和权限）。分发可刷新Token是可选的，如果认证服务器分发可刷新令牌，那么当分发访问 token的时候就会将可刷新令牌包括进去。

```
A refresh token is a string representing the authorization granted to the
client by the resource owner.  The string is usually opaque to the client.
The token denotes an identifier used to retrieve the authorization
information.  Unlike access tokens, refresh tokens are intended for use only
with authorization servers and are never sent to resource servers.
```

可刷新token是一个字符串，代表资源服务器对客户端的授权。该字符串对客户端是公开的的。token代表一个标示符，用于接收授权信息。与访问令牌不同的是，可刷新令牌只被用于认证服务器，而不会发送给资源服务器。发送给资源服务器的始终是访问 token。

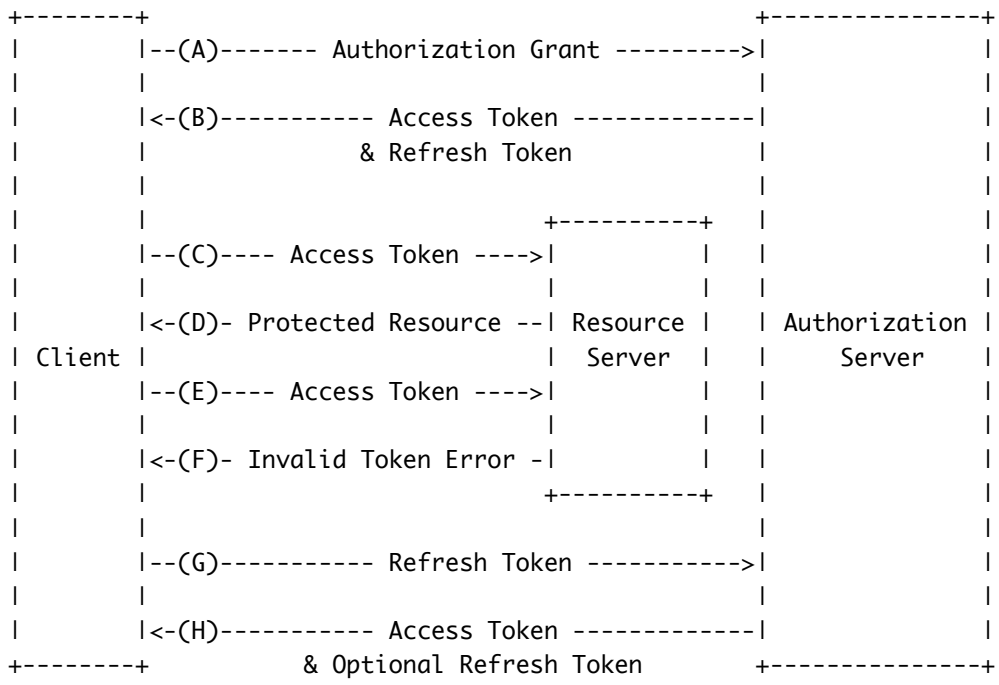

Figure 2: Refreshing an Expired Access Token

可刷新令牌的处理流程:

```
(A)  The client requests an access token by authenticating with the
     authorization server and presenting an authorization grant.

(B)  The authorization server authenticates the client and validates
     the authorization grant, and if valid, issues an access token
     and a refresh token.

(C)  The client makes a protected resource request to the resource
     server by presenting the access token.

(D)  The resource server validates the access token, and if valid,
     serves the request.

(E)  Steps (C) and (D) repeat until the access token expires.  If the
     client knows the access token expired, it skips to step (G);
     otherwise, it makes another protected resource request.

(F)  Since the access token is invalid, the resource server returns
     an invalid token error.

(G)  The client requests a new access token by authenticating with
     the authorization server and presenting the refresh token.  The
     client authentication requirements are based on the client type
     and on the authorization server policies.

(H)  The authorization server authenticates the client and validates
     the refresh token, and if valid, issues a new access token (and,
     optionally, a new refresh token).

Steps (C), (D), (E), and (F) are outside the scope of this
specification, as described in Section 7.
```

(A) 客户端访问认证服务器进行验证，通过提供提供授权认可，以请求访问token。

(B) 认证服务器验证客户端，并且验证授权认可（四种类型中的一种），如果有效则分发一个访问token和一个可刷新token。

(C) 通过提供访问 token，客户端可向资源服务器请求受保护资源。

(D) 资源服务器验证访问 token，如果有效，就响应请求。

(E) 步骤（C）和（D）重复进行，直到访问token过期。如果客户端知道访问token过期，就直接跳到步骤（G），否则它还会请求受保护资源。

(F) 由于访问token失效，资源服务器返回无效token错误。

(G) 客户端请求一个新的访问 token，通过向认证服务器提供可刷新 token 并进行验证。对客户端的验证条件是基于客户端的类型和认证服务器的策略。

(H) 认证服务器验证客户端和可刷新token，如果有效就分发一个新的访问token（也可以再发

送一个新的可刷新令牌）

2. 客户端注册(Client Registrantion)

```
Before initiating the protocol, the client registers with the authorization
server.  The means through which the client registers with the authorization
server are beyond the scope of this specification but typically involve end-
user interaction with an HTML registration form.
```

在初始化这个协议之前客户端需要通过授权服务器注册. 客户端通过授权服务器注册的方法已经超出了这篇协议的讨论范围, 但是一般来说是通过HTML注册表单来交互的.

```
Client registration does not require a direct interaction between the client
and the authorization server.  When supported by the authorization server,
registration can rely on other means for establishing trust and obtaining
the required client properties (e.g., redirection URI, client type).  For
example, registration can be accomplished using a self-issued or third-
party-issued assertion, or by the authorization server performing client
discovery using a trusted channel.
```

客户端注册并不需要一个客户端和授权服务器的一个直接的交互.只要授权服务器支持, 也可以通过其他的一些方法来建立信任或者得到客户端所需要的属性(比如说, 重定义URI, 客户端类型). 比如说, 注册可以使用自己发行的或者第三方发行的办法来完成, 或者通过授权服务器使用一条可信任的通道执行客户端发现.

```
When registering a client, the client developer SHALL:
o  specify the client type as described in Section 2.1,

o  provide its client redirection URIs as described in Section 3.1.2,
   and

o  include any other information required by the authorization server
   (e.g., application name, website, description, logo image, the
   acceptance of legal terms).
```

在注册一个客户端的时候, 客户端开发者应该:

o 指定在2.1节中描述的客户端类型,

o 提供开发者的在3.1.2节中描述的客户端的重定向URL

o 包含任何授权服务器所需要的其他信息(比如说, 应用程序名称, 站点, 描述, logo图片, 可接受的合法条例).

2.1. 客户端类型(Client Types)

```
OAuth defines two client types, based on their ability to authenticate
securely with the authorization server (i.e., ability to maintain the
confidentiality of their client credentials):
```

基于客户端通过授权服务器鉴定的安全性能力, OAuth定义了两种客户端类型(比如说, 维持他们客户端凭证的机密性的能力):

```
confidential
  Clients capable of maintaining the confidentiality of their
  credentials (e.g., client implemented on a secure server with
  restricted access to the client credentials), or capable of secure
  client authentication using other means.
```

机密的

客户端有能力维持他们凭证的机密性(比如说, 客户端通过受到保护的方式访问客户端凭证工作在安全的客户端), 或者有能力使用其他方式保证客户端安全.

```
public
  Clients incapable of maintaining the confidentiality of their
  credentials (e.g., clients executing on the device used by the
  resource owner, such as an installed native application or a web
  browser-based application), and incapable of secure client
  authentication via any other means.
```

公共的

客户端没有能力维持他们凭证的机密性(比如说, 资源拥有着使用的执行在设备上的客户端, 比如说一个需要安装的本地应用或者一个基于浏览器的web应用), 并且没有能力使用其他方式来保证客户端的安全性.

```
The client type designation is based on the authorization server's
definition of secure authentication and its acceptable exposure levels of
client credentials.  The authorization server SHOULD NOT make assumptions
about the client type.
```

客户端的指定是基于授权服务器对安全认证的定义以及对于客户端凭证可接受的级别. 授权服务器不应该假定客户端类型.

```
A client may be implemented as a distributed set of components, each with a
different client type and security context (e.g., a distributed client with
both a confidential server-based component and a public browser-based
component).  If the authorization server does not provide support for such
clients or does not provide guidance with regard to their registration, the
client SHOULD register each component as a separate client.
```

一个客户端可能是有一系列的分布式组件合作执行的, 每一个组件可能会有不同的客户端类型和安全内容(比如说, 分布式的客户端同时拥有一个机密的基于服务器的组件和一个公共的基于浏览器的组件).如果授权服务器不支持这样的客户端或者不提供这种类型的注册向导, 这样的客户端就只能每个组件注册一个不同的客户端了.

```
 This specification has been designed around the following client profiles:
```

这篇规范大致设计了如下客户端类型:

```
 web application
  A web application is a confidential client running on a web
  server.  Resource owners access the client via an HTML user
  interface rendered in a user-agent on the device used by the
  resource owner.  The client credentials as well as any access
  token issued to the client are stored on the web server and are
  not exposed to or accessible by the resource owner.
```

web应用程序

一个web应用程序是一个安全的客户端运行在web服务器上. 资源拥有着访问这些客户端通过资源拥有着持有的HTML用户接口在用户终端设备上渲染.客户端凭证同时也作为任何访问令牌部署到客户端的时候是存储在web浏览器中的, 这样就不容易暴露给资源拥有着或者被资源拥有着轻易取出.

```
user-agent-based application
  A user-agent-based application is a public client in which the
  client code is downloaded from a web server and executes within a
  user-agent (e.g., web browser) on the device used by the resource
  owner.  Protocol data and credentials are easily accessible (and
  often visible) to the resource owner.  Since such applications
  reside within the user-agent, they can make seamless use of the
  user-agent capabilities when requesting authorization.
```

基于用户代理的应用程序

一个基于用户代理的应用程序是指一个公共的客户端, 该客户端的代码是从web服务器上下载下来在用户终端中执行(比如说, 浏览器). 协议数据和凭证可以被轻易的访问(并且常常是可见的)

对于资源拥有者.因此像这样的应用程序存在于用户终端里面, 当请求授权的时候他们有完全控制用户终端的能力.

```
native application
  A native application is a public client installed and executed on
  the device used by the resource owner.  Protocol data and
  credentials are accessible to the resource owner.  It is assumed
  that any client authentication credentials included in the
  application can be extracted.  On the other hand, dynamically
  issued credentials such as access tokens or refresh tokens can
  receive an acceptable level of protection.  At a minimum, these
  credentials are protected from hostile servers with which the
  application may interact.  On some platforms, these credentials
  might be protected from other applications residing on the same
  device.
```

本地应用程序

一个本地应用程序是一个在资源拥有者的设备上安装和执行的公共的客户端.资源拥有者可以获取协议数据和凭证.假设任何存在于客户端的客户端认证凭证都可以被提取出来.另一方面, 动态的发布凭证比如说访问令牌或者可刷新令牌可以将安全保护级别提升到一个可以接受的层度.至少这些凭证可以防止与应用程序交互的敌对服务器. 在一些平台上, 这些凭证可能被同一个设备上的其他应用程序保护.

2.3. 客户端标示符(Client Authentication)

```
If the client type is confidential, the client and authorization server
establish a client authentication method suitable for the security
requirements of the authorization server.  The authorization server MAY
accept any form of client authentication meeting its security requirements.
```

如果客户端类型是机密型的, 客户端和授权服务器确定了一个满足授权服务器安全需求的客户端认证方法. 授权服务器可以接受任何形式的客户端授权请求来达到他的安全需求.

```
Confidential clients are typically issued (or establish) a set of client
credentials used for authenticating with the authorization server (e.g.,
password, public/private key pair).
```

授权服务器会发布一系列的安全证书给机密型的客户单用于与授权服务器的认证(比如说, 密码, 公共/私有的密匙对).