Version 1.0

2018-JuL-22



OAuth 2.0

Homework

Presenter: Arturo Sanchez

Mobile Apps Company

3301 Windy Ridge Parkway, Suite 400, Atlanta, GA 30339

# **OAuth 2.0**

## Introduction

### Definition

OAuth is an open standard for access delegation, commonly used as a way for Internet users to grant websites or applications access to their information on other websites but without giving them the passwords. This mechanism is used by companies such as Amazon, Google, Facebook, Microsoft and Twitter to permit the users to share information about their accounts with third party applications or websites.

OAuth 2 provides authorization flows for web and desktop applications, and mobile devices.

### Introduction

Generally, OAuth provides to clients a "secure delegated access" to server resources on behalf of a resource owner. It specifies a process for resource owners to authorize third-party access to their server resources without sharing their credentials. Designed specifically to work with Hypertext Transfer Protocol (HTTP), OAuth essentially allows access tokens to be issued to third-party clients by an authorization server, with the approval of the resource owner. The third party then uses the access token to access the protected resources hosted by the resource server.

OAuth is a service that is complementary to and distinct from OpenID. OAuth is also distinct from OATH, which is a reference architecture for authentication, not a standard for authorization. However, OAuth is directly related to OpenID Connect (OIDC) since OIDC is an authentication layer built on top of OAuth 2.0. OAuth is also distinct from XACML, which is an authorization policy standard. OAuth can be used in conjunction with XACML where OAuth is used for ownership consent and access delegation whereas XACML is used to define the authorization policies (e.g. managers can view documents in their region).

## How does it work?

### Roles

OAuth defines four roles:

#### Resource Owner: User

The resource owner is the user who authorizes an application to access their account. The application's access to the user's account is limited to the "scope" of the authorization granted (e.g. read or write access).

#### Resource / Authorization Server: API

The resource server hosts the protected user accounts, and the authorization server verifies the identity of the user then issues access tokens to the application.

From an application developer's point of view, a service's API fulfills both the resource and authorization server roles.

#### Client: Application

The client is the application that wants to access the user's account. Before it may do so, it must be authorized by the user, and the authorization must be validated by the API.

### Diagram



Explanation of the steps in the diagram:

1. The *application* requests authorization to access service resources from the *user*
2. If the *user* authorized the request, the *application* receives an authorization grant
3. The *application* requests an access token from the *authorization server* (API) by presenting authentication of its own identity, and the authorization grant
4. If the application identity is authenticated and the authorization grant is valid, the *authorization server* (API) issues an access token to the application. Authorization is complete.
5. The *application* requests the resource from the *resource server* (API) and presents the access token for authentication
6. If the access token is valid, the *resource server* (API) serves the resource to the *application*

The actual flow of this process will differ depending on the authorization grant type in use, but this is the general idea.

### Application Registration

Before using OAuth with your application, you must register your application with the service. This is done through a registration form in the "developer" or "API" portion of the service's website, where you will provide the following information (and probably details about your application):

* Application Name
* Application Website
* Redirect URI or Callback URL

The redirect URI is where the service will redirect the user after they authorize (or deny) your application, and therefore the part of your application that will handle authorization codes or access tokens.

### Client ID and Client Secret

Once your application is registered, the service will issue "client credentials" in the form of a client identifier and a client secret. The Client ID is a publicly exposed string that is used by the service API to identify the application, and is also used to build authorization URLs that are presented to users. The Client Secret is used to authenticate the identity of the application to the service API when the application requests to access a user's account, and must be kept private between the application and the API.

### Authorization Grant

In the Abstract Protocol Flow, the first four steps cover obtaining an authorization grant and access token. The authorization grant type depends on the method used by the application to request authorization, and the grant types supported by the API. OAuth 2 defines four grant types, each of which is useful in different cases:

* **Authorization Code**: used with server-side Applications
* **Implicit**: used with Mobile Apps or Web Applications (applications that run on the user's device)
* **Resource Owner Password Credentials**: used with trusted Applications, such as those owned by the service itself
* **Client Credentials**: used with Applications API access

### Grant Type: Authorization Code

The **authorization code** grant type is the most commonly used because it is optimized for server-side applications, where source code is not publicly exposed, and Client Secret confidentiality can be maintained. This is a redirection-based flow, which means that the application must be capable of interacting with the user-agent (i.e. the user's web browser) and receiving API authorization codes that are routed through the user-agent.

