AUTODESK **UNIVERSITY** 2014

OAuth 1.0 Versus OAuth 2.0 and Use Case

Cyrille Fauvel - Autodesk Developer Network Program Manager, Autodesk
Philippe Leefsma, Cloud Evangelist, Autodesk

SD5752 Many services such as Facebook, GitHub, and Google have already deployed OAuth 2.0 servers. The OAuth 2.0 spec leaves many decisions up to the implementer. Instead of describing the possible decisions that need to be made to successfully implement OAuth 2.0, this lecture will explain most of the appropriate decisions to make for most implementations. This lecture is an attempt to explain OAuth 2.0 in a simplified format to help developers and service providers implement the protocol. Attendees will also discover how they can use other people's OAuth servers instead of implementing their own.

Learning Objectives

At the end of this class, you will be able to:

- Understand differences between OAuth versions
- Understand requirements for implementing your own OAuth server
- Learn why you should use OAuth and how to use OAuth when exposing an API or using a third-party API
- Learn how to use third-party OAuth server in your infrastructure

About the Speaker

Cyrille Fauvel has been with Autodesk, Inc., since 1993, and he has been focusing on providing programming support, consulting, training, and evangelism to external developers. Cyrille is currently manager of Autodesk Developer Network (ADN) Sparks (or ADN Media & Entertainment), the worldwide team of API gurus that provide technical services through the Autodesk Developer Network.
http://adndevblog.typepad.com/cloud_and_mobile/

http://around-the-corner.typepad.com/ - *cyrille.fauvel@autodesk.com*

Introduction to oAuth

OAuth is an open standard for authorization. OAuth provides client applications 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

- Delegated Authorization - Access user's resources with user credential from a 3rd party service.
- Oauth is authorization not authentication
    - For Authentication take a look to OpenID Connect

http://en.wikipedia.org/wiki/OAuth

Rough Timeline of WEB technologies

- 2006 – Twitter OpenID
- 2007 – oAuth 1.0 (RFC 5849)
- 2008 - IETF normalization started in 2008
- 2009 – oAuth 1.0a
- 2010 – WRAP (Web Resource Authorization Profiles) proposed by Microsoft, Yahoo! and Google
- 2012 – oAuth 2.0 (RFC 6749)
  Bearer Token (RFC 6750)

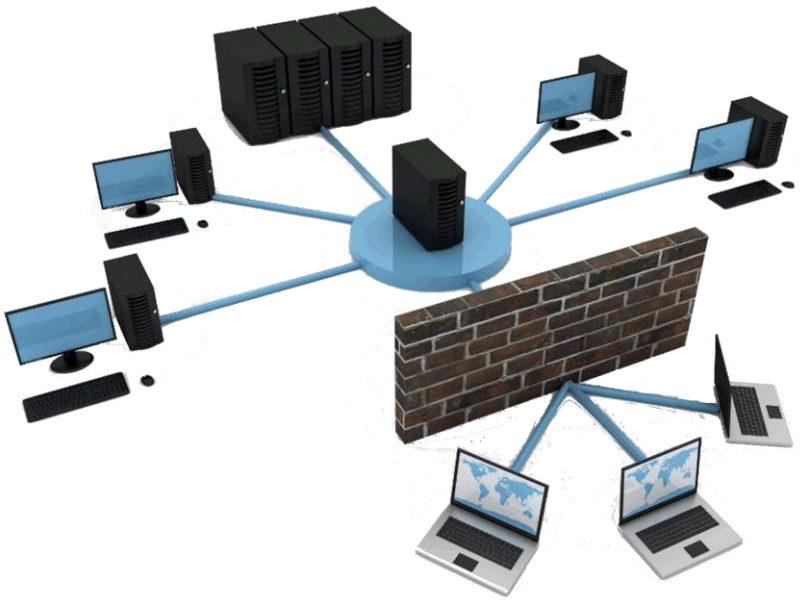
oAuth 1.0 vs. 2.0

oAuth 1.0(a) and oAuth 2.0 are different – not compatible. While oAuth 1.0(a) works on http and requires a signature to validate the call, oAuth2.0 works on https and relies on SSL (Secure Socket Layer) for encrypting the data.

In terms of simplicity, OAuth 2.0 does away with the need to digitally sign the tokens needed for authentication. Instead it encourages the use SSL (Secure Socket Layer) encrypted communications over HTTP (Hypertext Transfer Protocol), a combination called HTTPS. Most seasoned Web developers already understand how to work with HTTPS.

The fact you are using a secure channel doesn't mean the entity on the other side is good. It just means that no one else can listen in on it. If a client sends its bearer token to the wrong place, even over HTTPS, it's game over. oAuth 1.0a, uses http, so everyone can listen, but as parameters are signed for each calls, nobody can fake being you and call the end point. So oAuth 2.0 is much simpler for clients, while oAuth 1.0a requires you to calculate the signature (much harder). But on the server side, oAuth 1.0a is much simpler and free, while oAuth 2.0 requires IT guys to implement it and you to buy a certificate.

The new security stack for modern applications



Here everything is secured as long you are behind the wall – It is a local network with desktops. No data will go out. Later, we introduced laptops, and the need to access data from outside the office. Originally, a direct remote connection was used but not scalable.

WS* -

WS-SecureConversation

WS-Federation - http://en.wikipedia.org/wiki/WS-Federation

WS-Authorization

WS-Policy

WS-Trust

WS-Privacy

All managed by the enterprise security infrastructure. No trust issue as user and machine are all from the enterprise (user is somewhat trusted).

Very quickly, IT opened the network and put a firewall + security to protect the company network via identified protocols. Issue is B2b (Business 2 Business) introduced new protocols (SOAP / WS*) / (XML / SAML). Both side are controlled, but hard to implement.

The mobile Revolution

Game changer – when it came out, it did not care about enterprise at all – it was designed for consumers. But everyone now wants to use mobile devices for work as well (remember there more mobile devices than computer nowadays, and who in this room does not have one?).

Increasingly popular and business critical. Unfortunately, they do not support WS* protocols, but good at working with http(s).

Why do you need oAuth?

If you control both side (the client application running on your device/desktop and the service on your sever), you don't.

But what happens if the application is written by someone else? How much trust can you put on the application accessing resources on your back-end (or the other way around - you need to access someone else back-end). Even if you implement some security around http and https in your application or your server, how would you work with multiple 3rd party services, this is what oAuth is about.

Typical scenario

Actors

- Jane Doe (resource owner), Stores pictures on mypictures.com
- print.photos.net (client), Prints photos for user and collaborate with mypictures.com
- mypictures.com (the resource server), Exposes a protected API via oAuth

Simulation

- Jane wants to print the photos stored on mypictures.com using print.photos.net services.
    - she asks print.photos.net to do the job
- print.photos.net needs to access the resources from mypictures.com server
    - print.photos.net sends a request for a temporary request token
- mypictures.com validates the client request and sends it a temporary request token
    - mypictures.com returns a request token
- print.photos.net redirects Jane to mypictures.com for login
    - print.photos.net either redirects or gives the user the preformatted URL to sign on
- mypictures.com requests Jane to sign in using her credentials that authenticate her with the server and asks her approval to grant permission to the client to access her resources.
    - Jane logs on mypictures.com using login/password to approve giving access to client to access her resources
- print.photos.net is informed when Jane finishes granting authorization to the client
    - via a callback or …
- print.photos.net ask mypictures.com for an access token using its approved temporary request token
    - print.photos.net sends a request for a definitive access token

- mypictures.com validates the request and sends back an access token
    - mypictures.com returns an access token
- print.photos.net now access Janes photos from the mypictures.com server with this access token
    - print.photos.net does its work without knowing Jane's credential

Different scenario are also possible!

Examples

Demos and examples will demonstrate both:

- the ReCap Photo API (oAuth 1.0a) - https://github.com/ADN-DevTech/Autodesk-ReCap-Samples
- and the Autodesk 360 Viewer (oAuth 2.0) - https://github.com/Developer-Autodesk/autodesk-view-and-data-api-samples

Appendix: Resources

Autodesk Services samples

oAuth 1.0a - https://github.com/ADN-DevTech/AutodeskOAuthSamples

ReCap Photo API (oAuth 1.0a) - https://github.com/ADN-DevTech/Autodesk-ReCap-Samples

Autodesk 360 Viewer (oAuth 2.0) - https://github.com/Developer-Autodesk/autodesk-view-and-data-api-samples

AutoCAD i/o (oAuth 2.0) - https://github.com/Developer-Autodesk/AutoCAD.io

Servers Libraries

The Google PHP OAuth 1.0a Server library - http://code.google.com/p/oauth-php/

The Casablanca C++ RESTful SDK oAuth 1.0a and 2.0 server library - https://casablanca.codeplex.com/

Client Libraries

The Google PHP OAuth Client library - http://code.google.com/p/oauth-php/

The Guzzle PHP extension - https://github.com/guzzle/guzzle

The scribe-java library for Android - https://github.com/fernandezpablo85/scribe-java

The RestKit library for iOS and OSX - http://cocoapods.org/?q=restkit

C# library - RestSharp - http://restsharp.org/ - https://github.com/restsharp/RestSharp

The Casablanca C++ RESTful SDK oAuth 1.0a and 2.0 client library - https://casablanca.codeplex.com/

Servers examples

An oAuth 1.0a server example in Java - https://github.com/seratch/oauth-server-example

An oAuth 2.0 server example in C# - https://oauthserver.codeplex.com/

An oAuth 2.0 server example in PHP - https://github.com/thephpleague/oauth2-server

An oAuth 2.0 server example in Python - https://github.com/idan/oauthlib

For C++ see the Casablanca examples here - https://casablanca.codeplex.com/

Specifications

oAuth 1.0 (RFC 5849) - http://tools.ietf.org/html/rfc5849

oAuth 1.0a - http://oauth.net/core/1.0a/

oAuth 2.0 (RFC 6749) - http://tools.ietf.org/html/rfc6749

Bearer Token (RFC 6750) - http://tools.ietf.org/html/rfc6750