hueniverse

Menu

Skip to content

- Home
- Disclaimer
- OAuth
  - The OAuth 1.0 Guide
    - Introduction
    - History
    - Terminology
    - Specification Structure
    - Protocol Workflow
    - Security Framework
    - Authentication

Beginner's Guide to OAuth - Part I: Overview

With OAuth reaching its final draft (OAuth Core 1.0 Draft 4) last night, it is time for those of you new to the protocol to dive in and learn what it is all about. I have written in a previous post about the history behind OAuth, its use cases, and when it is (or isn't) applicable. People seems to like my metaphor of a valet key, which John Panzer rephrased "OAuth: Your valet key for the Web".

Beginner's Guide to OAuth - Part I

Introduction

This guide is intended for a technical audience with focus on implementation. I dedicate one section to the end-user perspective which is something I expect many others will address with mockups, user interface designs, best practices guides, and of course working services. To make the most out of this guide, keep the specification handy as I will be referencing it, walking you through the spec and adding color where needed. This guide does not replace the specification nor can it be used alone for implementation as it is incomplete.

End-User Benefits

OAuth allows you to share your private resources (photos, videos, contact list, bank accounts) stored on one site with another site without having to hand out your username and password. There are many reasons why one should not share their private credentials. Giving your email account password to a social network site so they can look up your friends is the same thing as going to dinner and giving your ATM card and PIN code to the waiter when it's time to pay. Any restaurant asking for your PIN code will go out of business, but when it comes to the web, users put themselves at risk sharing the same private information. OAuth to the rescue.

Both the valet key and ATM cards are good metaphors for OAuth from a user perspective. Instead of giving your ATM card and PIN code, the card can double as a credit card with a signature authorization. Just like your username and password provide full access to your resources, your ATM card and PIN code provide you with great control over your bank accounts - much more than just charging goods. But when you replace the PIN code with your signature, the card becomes very limited and can only be used for limited access.

Users don't care about protocols and standards - they care about better experience with enhanced privacy and security. This is exactly what OAuth sets to achieve. With web services on the rise, people expect their services to work together in order to accomplish something new. Instead of using a single site for all their online needs, users use one site for their photos, another for videos, another for email, and so on. No one site can do everything better. In order to enable this kind of integration, sites need to access the user resources from other sites, and these are often protected (private family photos, work documents, bank records). They need a key to get in.

The key used by users is usually a combination of username and password. This can be an OpenID or any other login credential. But this key is too powerful and unrestricted to share around. It also cannot be unshared once handed out except for changing it which will void access to every site, not just the one the user intends to block. OAuth addresses that by allowing users to hand out tokens instead. Each token grants access to a specific site (a video editing site) for specific resources (just videos from last weekend) and for a defined duration (the next 2 hours).

Unlike OpenID where users must do something first - get an OpenID identity they can use to sign-into sites - OAuth is completely transparent to the users. In many cases (if done right), the end-user will not know anything about OAuth, what it is or how it works. The user experience will be specific to the implementation of both the site requesting access and the one storing the resources, and adjusted to the device being used (web browser, mobile phone, PDA, set-top box).

A typical example offered by the spec (Appendix A) is when a user wants to print a photo stored on another site. The interaction goes something like this: the user signs into the printer website and place an order for prints. The printer website asks which photos to print and the user chooses the name of the site where her photos are stored (from the list of sites supported by the printer). The printer website sends the user to the photo site to grant access. At the photo site the user signs into her account and is asked if she really wants to share her photos with the printer. If she agrees, she is sent back to the printer site which can now access the photos. At no point did the user share her username and password with the printer site.

Scope

What is publicly known as 'OAuth' is really the 'OAuth Core 1.0' specification. The Core designation is used to stress that this is the skeleton other extensions and protocols can build upon. OAuth Core 1.0 does NOT by itself provide many desired features such as automated discovery of endpoints, language support, support for XML-RPC and SOAP, standard definition of resource access, OpenID integration, a full range of signing algorithms, and many other great ideas posted to the OAuth group.

This was intentional and is viewed by the authors as a benefit. As the name implies, Core deals with the most fundamental aspects of the protocol:

- Establish a mechanism for exchanging a username and password for a token with defined rights (Section 6).
- Provide tools to protect these tokens (Section 9).

It is important to understand that security and privacy are not guaranteed by the protocol. In fact, OAuth by itself provides no privacy at all and depends on other protocols to accomplish that (such as SSL). With that said, OAuth can be implemented in a very secure manner and the specification includes a good amount of security considerations to take into account when working with sensitive resources. Just like using passwords together with usernames to gain access, sites will use tokens together with secrets to access resources. And just like passwords, secrets must be protected.

Specification Structure

OAuth Core 1.0 includes 13 sections which are ordered in a way that allows a single pass through the spec without the need to go back and forth many times. If you want to dive right in, start with Appendix A, then read sections 3, 4.1, 6, 7, and 5.4. Read sections 5 and 9 when you are ready to implement.

- Section 1 – list of authors.
- Section 2 – general notations used by the spec.
- Section 3 – definitions of important terms used throughout the spec. While most are simple to understand, it is important to read this section a couple of times as it sets the framework for the rest of the document.
- Section 4 – protocol preparations. Before using OAuth, sites must follow a few steps to prepare for the protocol. The spec does not specify how this is done but does provide guidelines as to what information should be documented and established before the first OAuth request is made.
- Section 5 – implementation details on formatting parameters and interaction with the HTTP protocol.
- Section 6 – defines the mechanism for exchanging user credentials with a token. It is the heart of the specification and describes the entire flow including user interaction.
- Section 7 – defines how tokens are actually used to access resources. This short section is where most of OAuth takes place.
- Section 8 – technical details about creating nonce and timestamp values. Note that OAuth definition of timestamps does not necessarily means real time is used.
- Section 9 – while section 6 covers the first goal of the protocol, to exchange user credentials for a token, this section defines tools to protect the token from abuse. The signature process provides a working prototype and support for future extensions.
- Section 10 – suggested HTTP response codes. OAuth Core leaves much open for individual implementations.
- Appendix A – a complete example for sections 4 to 9. This is a good place to start reading if you intend to implement OAuth. It lets you dig right into the actual requests and see how they work.
- Appendix B – is an incomplete (as is true for most security papers) list of security considerations. It is absolutely critical that any OAuth implementer reads this thoroughly to understand the risks involved. Deciding on how to

address this list is up to each implementation and depends on needs.
- Section 11 – list of references and links to external documents.

Definitions

Section 3 contains definitions to fundamental protocol concepts referenced throughout the spec. Because understanding OAuth depends on these terms, they deserve some explanation:

- Service Provider – the Service Provider controls all aspects of the OAuth implementation. The Service Provider is the term used to describe the website or web-service where the restricted resources are located. It can be a photo sharing site where users keep albums, an online bank service, a microblogging site, or any other service where 'user's private stuff' is kept. OAuth does not mandate that the Service Provider will also be the identity provider which means the Service Provider can use its own usernames and passwords to authenticate users, or use other systems such as OpenID.
- User – the user is why OAuth exists and without users, there is no need for OAuth. The users have 'stuff' they don't want to make public on the Service Provider, but they do want to share it with another site. In OAuth, the protocol stops without manual interaction with the user at least once to receive permission to grant access.
- Consumer – this is a fancy name for an application trying to access the User's resources. This can be a website, a desktop program, a mobile device, a set-top box, or anything else connected to the web. The Consumer is the one getting permission to access resources and the Consumer is where the useful part of OAuth happens. OAuth defines 'Consumer Developer' as the entity writing code to interact with the Service Provider. 'Consumer Key' and 'Consumer Secret' will be explained later.
- Protected Resources: the 'stuff' OAuth protects and allow access to. This can be data (photos, documents, contacts), activities (posting blog item, transferring funds) or any URL with a need for access restrictions.
- Tokens – are used instead of User credentials to access resources. A Token is generally a random string of letters and numbers (but not limited to) that is unique, hard to guess, and paired with a Secret to protect the Token from being abused. OAuth defines two different types of Tokens: Request and Access. This are explained later in greater details.

Share this:

- Twitter
- Facebook
- Google
-

Like this:

Like Loading...

Related

October 4, 2007Eran Hammer OAuth, tutorial

Post navigation

← Twitter is More than Just Twitter
OpenID Makes Close Better →

17 thoughts on "Beginner's Guide to OAuth - Part I: Overview"

1. 
Ganesh says:
October 16, 2009 at 5:09 am

Very informative. Just the kind of information I was looking for. My brain however exploded trying to make sense of "With that said, OAuth can be implemented in a very secure manner and the specification includes a good amount of security considerations to take include account when working with sensitive resources." There probably is a grammatical mistake that I cannot lay finger or I have been working too much and my brain's damaged!

   - 
Eran Hammer-Lahav says:
October 16, 2009 at 8:58 am

Thanks for the correction. 'include' should have been 'into'.

2. 
Ganesh says:
October 16, 2009 at 5:09 am

Oh I forgot! Thanks for the write up :)

3. 
Senthil says:
October 23, 2009 at 9:19 am

Hi,
I like to know, what are the testing tools availble for Oauth/Restful.
Please let me know the same. It would be helpful.

Going through the documentation, it was so greatful in understanding. Very much thankful.

Eran Hammer-Lahav says:
October 23, 2009 at 9:37 am

http://wiki.oauth.net/ is a good place to look.

Leonid says:
December 1, 2010 at 7:27 am

Is there any opensource c# client libraries?

Regards,

Eran Hammer-Lahav says:
December 1, 2010 at 4:32 pm

Take a look at http://oauth.net/code/

4. 

rob ganly says:
October 27, 2009 at 7:10 am

good stuff! it seems that oauth is finally gaining momentum (since this was written 2 years ago). anyone know if there's a dedicated book in progress/planned?

it seems that documentation and detailed technical examples are still relatively scarce considering oauth is becoming such a big deal… hopefully that'll change too.

thanks,

rob ganly

5. 

danimataonrails says:
October 27, 2009 at 5:25 pm

Thanks for writing this down and sharing it.

6. 
JD Fulp says:
November 7, 2009 at 10:40 am

Have only skimmed this OAuth thing so far, but…. it looks like it's doing the same thing that SAML touts to do. For those familiar with both: where OAuth speaks "token", SAML would speak "assertion". Both provide no "native" security (confidentiality or authentication of exchanged data) and rely upon other services (SSL, PKI, VPN tunneling) to protect exchanged data.

Can anyone provide a quick compare/contrast between OAuth and SAML?

Thanks all…

JD

7. 
Kelvin says:
November 16, 2009 at 11:51 pm

any advise for implementing oauth with soap (wcf services)? thanks.

   - 
   Eran Hammer-Lahav says:
   November 19, 2009 at 10:12 am

   I am not aware of the two combined but it should not be too hard if you use the HTTP header to sent the authentication parameters.

8. 
Alex Penny says:
September 13, 2010 at 12:55 pm

Thanks for the info. I have been working with some API's but never really looked into OAuth

9. 
Monica says:
February 1, 2011 at 11:15 pm

This website provides very useful information

10. 
Steven Peterson says:
April 7, 2011 at 11:11 am

Thanks for such a great summary. I am left with the understanding that OAuth requires that a Consumer application and the Service Provider have a pre-existing relationship in which they have exchanged the shared secret — is this true? In other words, does OAuth allow for the situations in which the user, can direct a consumer application to the user's service provider to collect information, when the consumer application previously had no contact with the service provider?

    - 
Eran Hammer-Lahav says:
May 10, 2011 at 7:58 am

This is not typical at this time, but it is possible if the provider offers some method of using OAuth without client credentials or with fixed credentials. However, because we don't have discovery yet, even that has to be hardcoded somewhere in the client.

11. 
sanshi says:
June 18, 2011 at 7:51 am

Thanks for your great article, this is the Chinese version:
http://www.cnblogs.com/sanshi/archive/2011/06/17/2083670.html

Comments are closed.

Recent Posts

- On Securing Web Session Ids
- Introducing chairo, a hapi.js Microservices Plugin
- The Best Kept Secret in the Node Community
- On Leaving Walmart
- The Node Version Dilemma
- Why I Do Not Support a Node Foundation
- Notes on Managing Remote Teams
- Before the Drama
- Wide Open (or, Are You In?)
- Thank You
- Performance at Rest
- Dear CEO (of a node-powered corporation)
- Names and Diversity
- Open Source ain't Charity

- Open Source Dickishness
- The Fallacy of Tiny Modules
- Don't Be a Bully
- #NodeDay Slides by Chris Carrasco
- On Being (Mentally) Well
- Speakers Creativity Budget

Categories

- Architecture
- Cartoons
- Discovery
- Featured
- Guest Writer
- Microblogging
- Node.js
- OAuth
- Open Web
- OpenID
- Opinions
- Personal
- Recap
- Security
- Sled
- Social Web
- Startup
- Sunday Reading
- Uncategorized
- WebFinger
- Work
- Work Culture
- XRD

Search site

Search Search … Search