Understanding OAuth

akana™
Powering the API Economy

akana
Powering the API Economy

Introduction

No one knows who first uttered the phrase, “Necessity is the mother of invention,” but it can be traced back as far as the 15th Century or even to Plato. It’s an old idea, but one that never loses its relevance. New challenges require new solutions. We are now witnessing such a moment with API security. The spectacular rise of sophisticated web applications, mobile apps and unique business models has created a situation where traditional authorization mechanisms simply do not provide an effective solution for sharing data and federating identity amongst multiple trusted parties, such as Facebook and Twitter. The use-cases have evolved to scenarios wherein it is not the user himself or herself that is directly access the information, but delegating rights to other applications on the users’ behalf to access all or a subset of data.

It’s a simple problem with complicated implications. With a client/server architecture, the client software requests access to information controlled by the server. That assumption is that it’s abinary transaction involving the client (usually operated by one human being) and the server. This is simply not an assumption one can make any longer. Today, there are myriad use cases where one server must tap into one or more other servers before it can fulfill a request from a client, or where a client itself must communicate with multiple servers to complete an action. This creates a big authentication and authorization headache. For example, if a consumer wanted to print photos from a social media site at a photo printing service and pay with credit card rewards points, the user has to give permission for the printing service to access his social media photo album as well as his rewards account. The challenge is to do this without requiring the user to give his login credentials to the printing service.

OAuth arose out of the process of improving the OpenID standard at the Internet Engineering Task Force (IETF) to solve this problem of secure access to multiple systems on behalf of a single client. This paper illustrates how OAuth works using simple use cases. It delves into the history of OAuth and contrasts it to OpenID, a comparable but different method commonly used for authentication. Then, this paper takes a look at a more complex enterprise use case and discusses how an API management solution can help facilitate the effective use of OAuth in the context of corporate computing.

Contents

akana
Powering the API Economy

The Authorization Dilemma

Consider the following scenario: You subscribe to MyBucks, an online service and mobile app that enables you to see a single report of your portfolios at multiple brokerages. For example, MyBucks could show you your 401K balance at Brokerage A and your IRA balance at Brokerage B on one app screen. To function, MyBucks needs to be able to log into your personal accounts at each financial house where you have a brokerage account. This architecture is shown in Figure 1.

MyBucks could get your account information by asking you for your full log in details (usernames and passwords) and logging into your various brokerage accounts. However, this is not an optimal or wise arrangement. Giving that level of access to your personal financial accounts to a third party exposes you too much risk. It may not even suit MyBucks' branding goals. They probably don't want to appear as if they want too much of your private information.

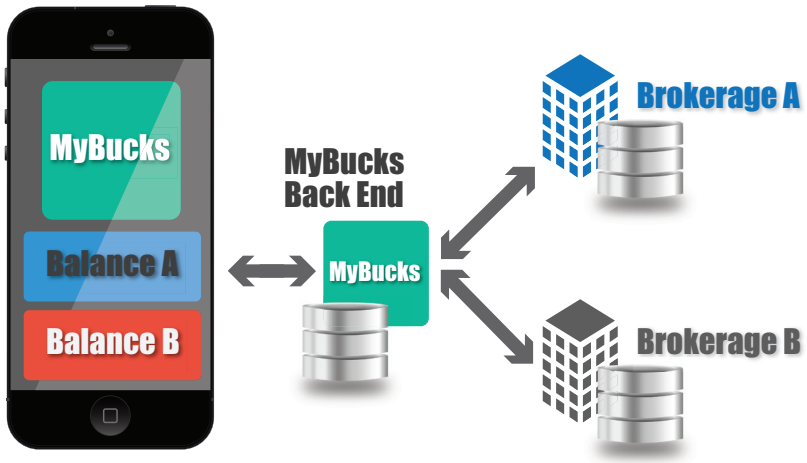

Figure 1 – The authorization dilemma facing many apps and websites: How can you give permission for an application to view some, but not all of your data held by a third party without revealing your log in credentials?

The Development of the OAuth Standard

The industry has been trying to solve the authorization dilemma for a number of years. In 2006, Blaine Cook, then serving as Lead Developer at Twitter, confronted it when he was working on Twitter's implementation of OpenID. OpenID did not provide for the delegation of API access. Cook, along with others in the industry that were struggling with the same issue, came together to develop the OAuth standard. The standard was published by the Internet Engineering Task Force (IETF) in 2010 (Version 1.0) and updated in 2012 (Version 2.0).

There's nothing wrong with OpenID. It was just created to solve a different problem. It exists in parallel with OAuth, but it's worth taking a look at how OpenID works to understand why it was necessary to bring OAuth into existence. OpenID enables you to use your login at one site, such as Google, to access another site. When you use OpenID, the site you are trying to log into asks for a confirmation of your identify from the site you reference. If you log in using Google as your Open ID reference, the destination site will receive a certificate from Google that authenticates your identity. There are several drawbacks to this approach:

- OpenID does authentication, but it is not set up to handle authorization. You can use OpenID to establish who you are, but the site that lets you in has no idea what you're entitled to see.
- It assumes that the person requesting access is the same person/entity that controls the requested information resource.
- OpenID requires the direct participation of the individual user, who has to know that he or she can use OpenID to access multiple sites without creating a log in for each one. This requirement has led to uneven adoption of OpenID. Many web users don't know that OpenID exists or if they do, what it does.

akana
Powering the API Economy

How OAuth Solves the Authorization Dilemma

OAuth was devised to solve the authorization challenge shown in Figure 1. How can MyBucks get access to your brokerage accounts, which are held by completely separate entities, without giving MyBucks full access to your brokerage accounts? You only want MyBucks to get your account balance. You don't want MyBucks to be able to transfer funds, make withdrawals, and so forth. The OpenID approach, which enables you to log into a site using credentials from a third party, such as Google or Facebook, has numerous limitations that make it far from ideal for the MyBucks use case. The OpenID approach would not be able to restrict the type of information that you might want to restrict MyBucks from having access to. OpenID by itself provides primarily an all or nothing approach.

OAuth is an open standard for authorization that enables an application to request access to third party systems on behalf of its users. OAuth makes this possible without the need to share sensitive personal login information with MyBucks. Rather, it creates a secure token that allows one system to access specific information and functionality from another system.

Figure 2 adds more detail to the MyBucks use case to show how OAuth manages the tricky authorization needed for the MyBucks application to function securely. User X requests his brokerage account balances from MyBucks. MyBucks then sends the brokerages a data request containing two unique OAuth tokens, each created expressly for User X at the two respective brokerages. The OAuth tokens themselves do not contain User X's personally identifying information. However, based on the trust relationship that has been previously established between MyBucks and each brokerage, the brokerage systems understand that the OAuth tokens are for User X. The OAuth setup and tokens can also be configured to permit MyBucks to have access to some, but not all, of User X's account information. The OAuth tokens are generated the first time MyBucks need to access information from a specific brokerage on behalf of the user. As part of this, MyBucks requests authorization from the User.

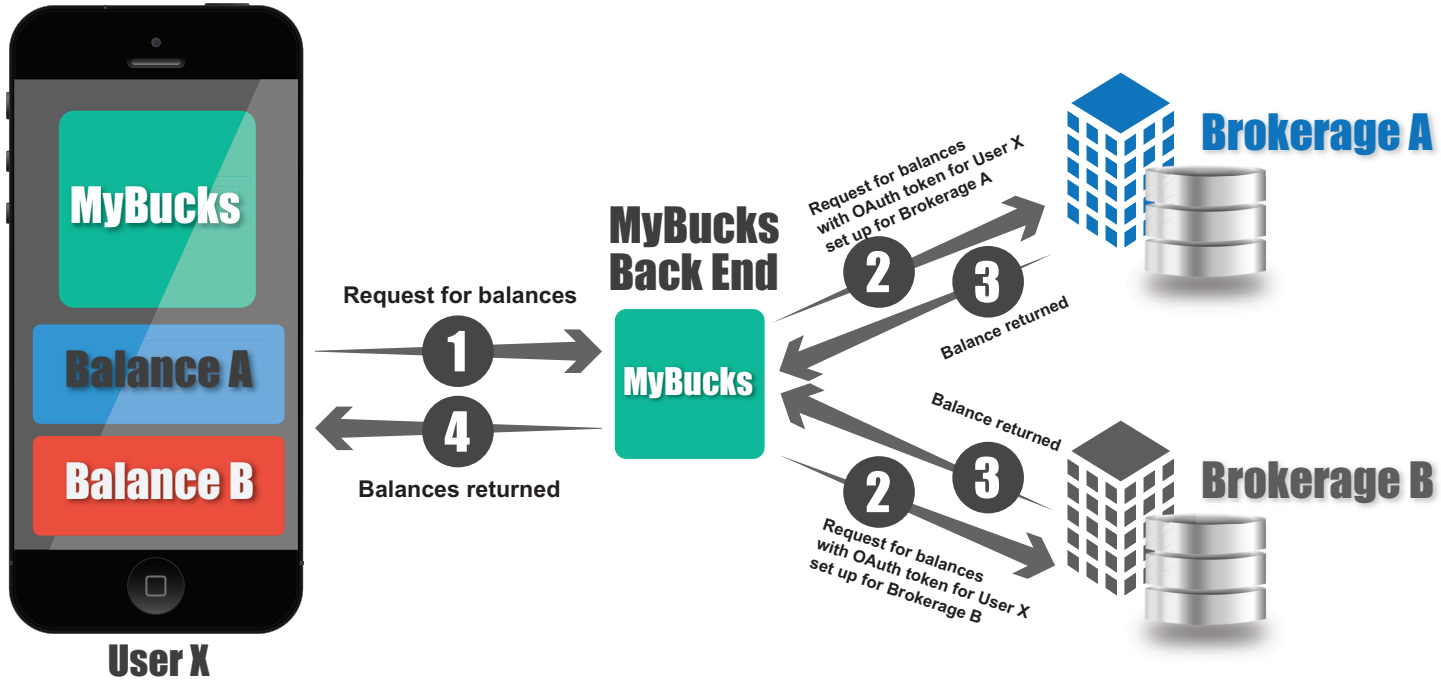

Figure 2 – The OAuth solution: An OAuth security token enables the data owner to grant limited access to another application without revealing identifying information.

akana
Powering the API Economy

There are several advantages to the OAuth approach. OAuth makes it possible for the owner of information resources, who is typically the end user, to grant access to those resources to any number of parties without revealing personal information. OAuth can also be set up for limited information access rights. The popular metaphor in use is to compare an OAuth token to a car's "valet key." Like the valet key that will turn a car on and open the door, but not the trunk, an OAuth token can grant access to a portion of a resource owner's data but not all.

Other OAuth use case examples include:

- **Photo print service needs access to your images** – You might store your images on a cloud-based service. A third party photo printing service needs access to your images. You log into the image server and authenticate yourself to it. That way, the image server can grant access to the photo printing service using OAuth.. With an OAuth token to identify you and authorize access to your photos, you can grant the photo printing service access to your images without having to share your log in credentials.

- **The government needs access to your smart electric meter** – Smart electric meters store information about power usage that can be valuable to the government. If the government wants to tap into to that information, OAuth can be used to grant access without having to trust the government with your personally identifying information or log in credentials.

- **A web site wants access to your social graph** – When a website wants access to your social graph on Facebook or your Twitter feed, it can use OAuth to establish the connection. Many popular sites use OAuth, including Amazon, Dropbox, Facebook, Twitter, Flickr, Google, Instagram, NetFlix and many others.

How OAuth Works

Getting OAuth working involves a multi-step process. As shown in Figure 3, the two entities that want to participate in the exchange of OAuth tokens must first establish trust. They do this by exchanging a "shared secret" that consists of Client IDs. When that trust has been established, it is possible for the requesting entity to set up an OAuth client. The OAuth client sends information requests along with a unique OAuth token that has been created just for the resource owner. The token grants access to data on a basis defined by the resource owner. The token does not contain the resource owners' user name and password.

In practical, day-to-day use, the OAuth process looks like this: A user logs into MyBucks to add Brokerage A's account to the MyBuck's app. The MyBucks back-end asks the user if he gives permission for MyBucks to access his account at Brokerage A. MyBucks will then present a login screen for Brokerage A, asking the user if MyBucks can access data on his behalf. The process is flowing through the MyBucks app but the user is interacting directly with Brokerage A's systems. After the permission has been granted, the Authorization Server will create an OAuth token that is unique to the user. The token can outlast the browser session and expire based on policy set by Brokerage A.

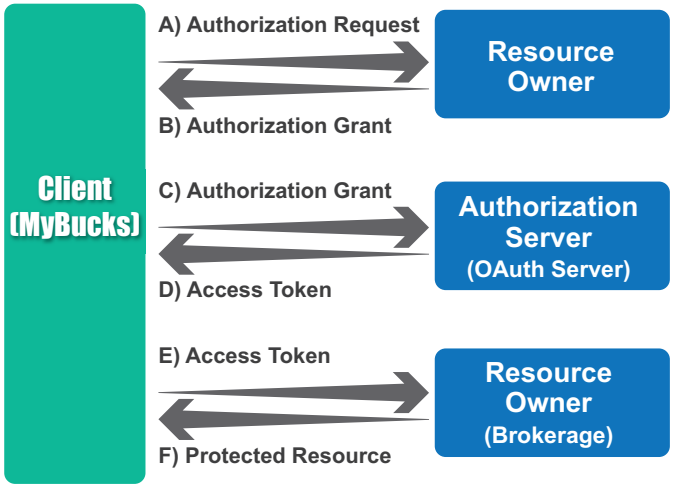

Figure 3 – The progression of an OAuth authorization grant. Initially, the client must request authorization from the resource owner. That authorization is then used by the client to get an OAuth access token from the OAuth server, which is in turn used by the client to get access to the resource owner's protected resource.

The Role of the API in the OAuth Transaction

Application Programming Interfaces (APIs) created using the Representational State Transfer (REST) protocol have become a nearly universal standard today for connecting mobile apps and websites with servers and third party systems. An API consists of libraries and protocols used as interfaces between software components. RESTful APIs (also referred to as "Web APIs") use simple, standards-based languages such as JavaScript Object Notation (JSON) and eXtensible Markup Language (XML) and transmit messages using HTTP. By convention, nearly all web APIs use the same structure and rely on a few accepted REST procedure calls, which are based on Hypertext Transfer Protocol (HTTP): GET, POST, PUT and DELETE.

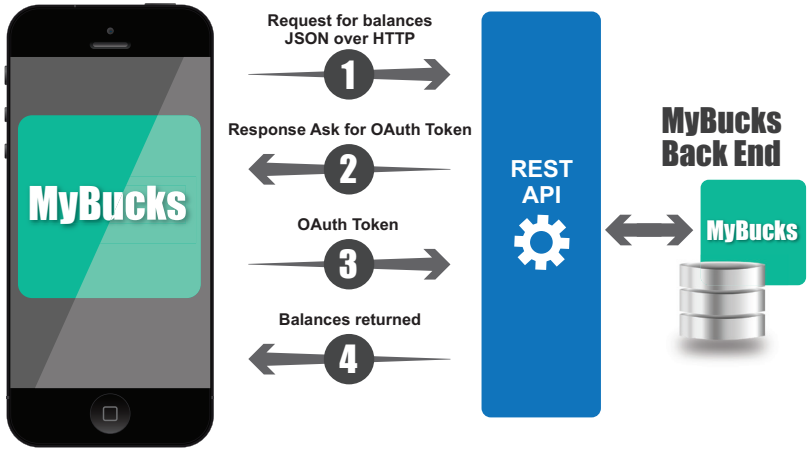

Figure 4 – A RESTful API serves as the communication gateway between a website and mobile apps. This common protocol has become a nearly universal standard for communication between websites, apps and other online systems, especially in the consumer technology segment.

Figure 4 shows the workings of the RESTful API in the MyBucks architecture. The request for balance information flows from the MyBucks app to the MyBucks back end servers. The request is written in JSON for the REST protocol and transported using HTTP. The MyBucks REST API is the

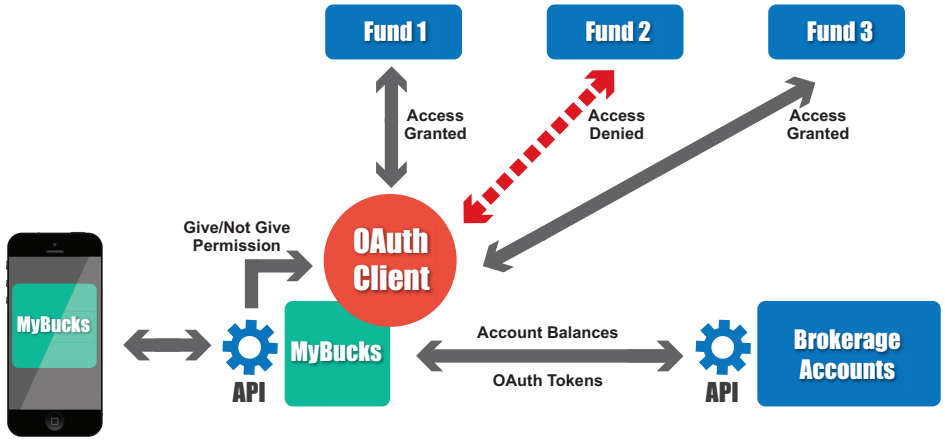

Figure 5 – An enterprise OAuth scenario: Multiple external parties want access to MyBucks user account information for research and permission-based marketing. OAuth can streamline the access granting or denying process.

communication switchboard for the entire MyBucks system. When the MyBucks back end receives the request from a user for one of the user's brokerage account balances, MyBucks requests that user's OAuth token. The user's app responds with the OAuth token that is sent back to the backend server. MyBucks then forwards the balance request to the brokerage house together with the OAuth token

RESTful APIs have grown in popularity as the proliferation of mobile devices and innovate web applications creates more demand for connectivity between end users and back-end systems. REST is simple, lightweight, and standards based – an essentially free way to integrate any number of unrelated application using the web. Originally the province of the consumer Internet, REST is rapidly becoming part of the corporate enterprise architecture. Businesses, eager to adopt REST to connect with new clients and partners, are increasingly configuring enterprise systems with REST APIs.

Where do RESTful APIs come from? Many are built from scratch using software development tools. However, as REST grows in popularity, most major software development toolsets are now including automated REST API creation capabilities. Packaged software products are also now beginning to ship pre-set REST APIs for simple deployment.

akana
Powering the API Economy

Enterprise OAuth Scenarios

OAuth was birthed in the consumer Internet, but it has many uses in the enterprise context. Figure 5 shows how the simple MyBucks use case can be expanded to include multiple enterprise partners. In this example, mutual funds want to access MyBucks' client portfolios in order to conduct a free investment analysis and make suggestions about new investment opportunities. MyBucks cannot simply grant them access. The individual client must have the chance to review the request for access and either grant or deny access. And, if the client wishes to grant access, MyBucks should ideally enable them to grant selective access, allowing the client to choose whether the fund can review some or all of the client's portfolio data. OAuth is able to handle the challenges raised by this use case.

After MyBucks has established trust by exchanging secret client IDs with each fund, the funds can ask each MyBucks client if they wish to grant them access to their investment portfolios. If the client agrees, the fund gets a unique OAuth grant it can use to request the portfolio data. This a great use of OAuth. The client gets the benefit of an investment review without having to give a relatively unknown entity its log in or identifying information..In addition, the funds can review all of MyBucks' clients' portfolios in aggregate, giving each client the advantage of an account overview they could never get on their own. The fund benefits because it is able to prospect for new clients. MyBucks benefits because it can offer a value-added service to its clients and perhaps earn fees for helping the funds sign up new clients. OAuth makes it possible.

Effective, Efficient Approaches to Enterprise OAuth

The enterprise OAuth use case begs the question: How can an organization keep track of multiple entities requesting and receiving OAuth tokens? That is, how can it be done without requiring a lot of person-hours and manual, error-prone processes? The short answer is that OAuth management should be folded into the broader security automation and governance platforms already in use. Akana has taken this approach, embracing OAuth as part of its overall API management solution.

OAuth management is a natural extension of the API management functionality. The Akana API management solution provisions API access selectively to app developers and manage APIs throughout the plan/build/run/share lifecycle of the API. It then lets users connect existing security systems with their OAuth servers, a capability that leads to further efficiency and effectiveness. For instance, if an organization uses Microsoft Active Directory or IBM Tivoli Identify Manager, the Akana OAuth server can federate OAuth tokens with the permissions already established within these central identity stores.

From a security perspective, the OAuth transaction is really just part of an overall trust relationship. An entity that is not trusted in general should not be able to request and receive OAuth tokens. This could easily happen if API versions are not managed, especially in today's enterprise API reality, where there may be scores of APIs in development and production, connecting to myriad third parties, at any given time.

It also makes sense to unify the management of individual operations enabled by the API with the selective access granted by the OAuth token. To understand what this means, consider that most APIs expose more than one type of back end functionality. A single API might

contain operations that make distinct types of information available to client applications. For example, a bank API might have operations for account balance query, transaction confirmation, stop payment, and so forth. The OAuth tokens issued against this API should ideally be able to specify precisely which operations are allowed and which are not – and, most importantly, this granular access control needs to be at least partly automated. If it isn't, the administrative burden on the OAuth process will be too high.

Conclusion

Enterprises need to share their own data, and the data of other organizations, freely and securely among Web applications, mobile devices, and through APIs. This is the purpose of OAuth: Easy access to and transmission of secure information. Akana's OAuth Server makes it possible to realize the purpose of OAuth in the enterprise context, making it efficient to use OAuth to access and transmit data in a reliable, governed infrastructure.

Appendix

The Akana OAuth Server

The Akana OAuth Server provides comprehensive, standards-based authentication and authorization capabilities to the enterprise, and includes:

- Integration with Most Enterprise Identity and Access Management Systems – includes LDAP, Active Directory, CA SiteMinder, Oracle Access Manager, IBM TAM, RSA ClearTrust and more.
- Open Specifications Support – includes support for the OpenID, OAuth 1.0a, OAuth 2.0 and SAML specifications along with a wide array of other authentication and authorization specifications.
- Authorization Management - a fully brandable user experience allowing businesses to integrate OAuth seamlessly into their existing applications.
- Built-in grant management – allows users to confidently make and manage their authorization decisions.
- Support for Multiple Providers – OAuth Server allows for provider support based on each one's own set of resource and grant definitions, allowing the enterprise to offer a range of different OAuth services for different applications and purposes.
- Interoperability – guaranteed interoperability with most common systems, based on extensive lab and field interoperability testing.
- Centrally Managed Access Control -OAuth Server allows users to centrally manage the applications they want to interact and share private data with. Security features include:
- Authentication Policy Options - Choose from a wide array of authentication schemes, standards and token types to ensure that only valid users and applications get access to your APIs.

akana
Powering the API Economy

- Enterprise OAuth - Use your existing enterprise security systems to create an OAuth authorization server so your users can manage access rights for their own data.
- Advanced Cryptography - Ensure the privacy of customer data with sophisticated encryption and signature capabilities. Mobile device users can seamlessly access data from multiple enterprise and cloud applications with the help of these features:
- Single-Sign-On for Mobile Applications - Use enterprise credentials for single-sign-on to native and browser-based applications without sharing passwords.
- Mobile Application Authorization - Centrally manage authorizations for mobile applications.
- API Platform Integration - Akana's OAuth Server offers deep integration with the company's Enterprise API Platform, which allows customers to extend enterprise security to their APIs:
- Resource to API Mapping - Define how OAuth resources map to APIs to enforce authorization decisions at the edge of your network.
- Choose OAuth Options in API Configuration - Allow customers to tailor the OAuth capabilities offered by the APIs for mobile, desktop, or browser applications, as well as specific security requirements.

About Akana

Akana is a leading provider of API Security and Management products that help businesses plan, build, run and share APIs, through comprehensive cloud and on-premise solutions that encompass API lifecycle, security, management and developer engagement. The world's largest companies including Bank of America, Pfizer, and Verizon use Akana solutions to transform their business.

For more information, please visit http://www.akana.com

Akana, API Gateway, Community Manager, Lifecycle Manager, Policy Manager, Portfolio Manager, Repository Manager, Service Manager, and SOLA are trademarks of Akana, Inc . All other product and company names herein may be trademarks and/or registered trademarks of their registered owners.

akana™

Powering the API Economy

Akana
12100 Wilshire Blvd, Suite 1800
Los Angeles, CA 90025

(866) 762-9876 | www.akana.com | info@akana.com