Research Report Oauth2



By Jonathan Kat

# Table of Contents

Contents

[Table of Contents 2](#_Toc152575529)

[Introduction 3](#_Toc152575530)

[What is OAuth2? 4](#_Toc152575531)

[History of OAuth 5](#_Toc152575532)

[Before OAuth 5](#_Toc152575533)

[The Beginning of OAuth 5](#_Toc152575534)

[Change to OAuth2 6](#_Toc152575535)

[Authentication vs Authorization 7](#_Toc152575536)

[How does OAuth2 work? 8](#_Toc152575537)

[OAuth vs OpenID 9](#_Toc152575538)

[Summary 10](#_Toc152575539)

[What is OAuth? 10](#_Toc152575540)

[How did OAuth start? 10](#_Toc152575541)

[What is the difference between Authentication and Authorization? 10](#_Toc152575542)

[How does OAuth2 work? 10](#_Toc152575543)

[Conclusion 11](#_Toc152575544)

[Sources 12](#_Toc152575545)

# Introduction

Whenever you are logging into Facebook, Spotify or Netflix. You probably want to login safely and securely, or you might want to login easily by pressing the “Login with Google” button. This is done by using OAuth2. You may think, what is OAuth2? How does it work? What does it do? Well I think exactly the same so I decided to research it. I want to know: **“What is the function of OAuth2?”**  
This is the main question I will be trying to answer.

To try and answer the main question I want to try and answer a few Sub-Questions along the way:

* What is OAuth?
* How did OAuth start?
* What is the difference between Authentication and Authorization?
* How does OAuth2 work?

# What is OAuth2?

OAuth2, which stands for “Open Authorization”, allows a website or an application to access resources hosted by other web applications on behalf of a user, or in other words, it is the “login with Google” or “login with Facebook” buttons you see on certain websites to login, without having to give your credentials to the place you want to login.   
People think when they press those buttons that the application gets sent the login credentials (username and password), but this is not the case. I will talk about how it works later on in the report.

In 2012 OAuth 2.0 replaced OAuth 1.0 and is now the industry standard for online authorization. OAuth2 provides consented access and restricts certain actions of what the client application can perform on resources on behalf of the user, without ever using the user’s password.  
Even though OAuth2 has the web as it’s main platform, the specification describes it is also able to handle this kind of access to other types of clients, like mobile applications and browser-based applications.

# A screenshot of a facebook account Description automatically generatedHistory of OAuth

## Before OAuth

Before OAuth first released in 2007, when you wanted to connect your Facebook account to your Gmail account, you had to provide Facebook with your Gmail password. In figure 1 it says that “Facebook will not store your password.” But this might as well be a lie and they could store and sell your password to third parties so they could get access to your gmail account.

Figure | Old facebook page to "Find Friends" who are already using Facebook

## The Beginning of OAuth

In November 2006, developer Blaine Cook was actively engaged in implementing OpenID for Twitter. During this period, he reached out to Chris Messina to explore the integration of OpenID with the Twitter API for authentication delegation. They convened with David Recordon and Larry Halff at an OpenID meeting to explore potential solutions. Larry sought a method to incorporate OpenID into Ma.gnolia's (now Gnolia) Dashboard Widgets. Following a thorough assessment of existing OpenID functionality and industry practices, they collectively concluded that there was no established open standard for API access delegation. These discussions persisted both online and offline for several months.

In April 2007, a Google group was established, comprising a small team of implementers tasked with formulating a proposal for an open protocol. This challenge, they discovered, was not exclusive to OpenID. Upon learning about the initiative, DeWitt Clinton from Google expressed his interest in endorsing the effort, even if only in a stakeholder capacity. By July 2007, the team had crafted an initial specification, and the group extended an invitation to anyone interested in contributing to the project. On October 3rd, 2007, the final draft of OAuth Core 1.0 was officially released.

## Change to OAuth2

In October of 2012, OAuth2 was released to replace OAuth 1.0. There were drastic changes beween 1.0 and 2.0. In many ways, OAuth2 is *less* secure, more complex and less prescriptive than version 1.0. The creators of version 2.0 focused on making OAuth more compatible and flexible between sites and devices. Regadless of intent, many of the original founders and supporters of OAuth stated that they did **not** support version 2.0

The changes in verison 2.0 were so significant that it’s not compatible with verison 1.0, and even different uses of version 2.0 may not work seamlessly with eachother. However, nothing stops a website from supporting both 1.0 and 2.0, although the creators of 2.0 released it with the intent to completely replace 1.0.

One of the biggest critiscisms of OAuth2 is that the standard does not directly define or support encrypton, signature or client verification on purpose. Instead, OAuth expects users to use an outside protection protocol like Transport Layer Security (TLS), to provide those features.

# Authentication vs Authorization

While Authentication and Authorization are often used interchangeably, they are fundamentally different things.

Simply said, Authentication is the process of verifying who a user is, while Authorization is the process of verifying what the user has access to.

A Real-world example: When you go to the airport, you go through security where they check your ID so they know who you are (Authentication). Then when boarding starts the flight attendant looks at your boarding pass so they can allow you to enter the plane (Authorization)

A black screen with white text

Description automatically generatedIn the table below you are able to see the different functions of both Authentication and Authorization.

Table | Difference between Authentication and Authorization. Source: [Auth0](https://auth0.com/docs/get-started/identity-fundamentals/authentication-and-authorization)

# How does OAuth2 work?

A screenshot of a phone

Description automatically generatedThere are 3 main roles that are used with OAuth.

1. User

The user is the user of the application. 99.9% of the time a human being

1. The Application (The Client)

This is what is being used. For example Spotify

1. The API (The Authorization Server)

The api is your “Login with …..” button. In figure 2, the example is “Login to Facebook”

Figure | OAuth 2.0 Roles. Source: [InterSystems Learning Services](https://www.youtube.com/watch?v=CPbvxxslDTU&t=203s)

In general there are 8 steps involved into authenticating an API call.

1. Client Registration

The client needs to register with the authorization server to obtain certain credentials (ID and Secret). A few examples of these authorization servers are Auth0 authorization server, Facebook authorization server and Google authorization server. These are just a few examples, but there are many more.

1. User Authorization

The client redirects the user to the authentication server where the user will be asked to log in and grant permission to the client.

1. Grant Authorization

When the user has granted authorization to the server, the server provides an authorization grant to the client. An authorization grant serves as proof that the user has been authorized.

1. Access Token Request

The client then uses the authorization grant and the client credentials (ID and Secret) to request an access token from the authorization server.

1. Issuing of Access Token

Now the server verifies the provided authorization grant and client credentials and return an access token to the client.

1. Make API Request

The client then enters the access token in the parameters of its API request to the backend. The access token serves as proof of authentication and authorization

1. Validate Token

The backend validates the access token to ensure that it is valid and authorized to call the backend. If the check fails, then the backend considers the token invalid and rejects the API request.

1. API Response

If the access token is valid, the backend processes the API request and returns the appropiate response to the client.

# OAuth in practice

OAuth is being used in many different ways. One of these ways is just simply, logging in. Lets say you want to login to Spotify, but you want to login with Facebook for convenience. Then you could press the “Login with Facebook” button and you make or login in Spotify with your Facebook. This also makes it so that you give Spotify access to certain data from Facebook, like profile picture and name, without having to give any of your sensitive credentials, like your password.

OAuth is not only used for logging into third party apps like Spotify, but it could also be used to log in to first party apps. Lets take google for example. It doesn’t matter if you try to login to YouTube, Google Maps or Gmail. The login button will always take you to accounts.google.com. Which makes the entire Google ecosystem easy and safe to use since all of your data will be going through 1 server, making it harder for externals to crack.

# OAuth vs OpenID

There are a couple of other security technologies that you might hear about in the same context as OAuth. I have mentioned it before and one of them is OpenID. At a base level, it is pretty simple to see the difference between the two. Remember how OAuth stands for Open Authorization? Well, OpenID is more about authentication, or how StackOverflow commenter [Jörg W Mittag](https://stackoverflow.com/questions/4230821/if-openid-is-dead-what-is-out-there-to-take-its-place/4230970#4230970) put it: “OpenID is for humans logging into Machines, OAuth is for machines loggin into machines on behalf of humans.”

OpenID was released in 2005 as a means to log into the then popular LiveJournal blogging site, but quickly spread to other sites. The idea, in the early days of Web 2.0, was that instead of having multiple logins for multiple websites, OpenID would serve as a single sign-on, essentially vouching for the identities of users. But in practice OpenID was not easy to implement for developers, and never became appealing for users. In 2011, [Wired declared](https://www.wired.com/2011/01/openid-the-webs-most-successful-failure/): “The main reason no one uses OpenID is because Facebook Connect does the same thing and does it better. Everyone knows what Facebook is and it’s much easier to understand that Facebook is handling your identity than some vague, unrecognized thing called OpenID.”

But that is not quite the end of the story for OpenID. In 2014, OpenID Connect was released. OpenID Connect reinvented OpenID as an authentication layer for OAuth. In this space, OpenID has found purpose, and the two technologies now complement eachother in many different ways.

# OAuth and OWASP

The Open Web Application Security Project (OWASP) is an open community dedicated to enabling organizations to develop, purchase, and maintain applications and APIs that can be trusted. This report is focussed on OAuth2 and logging into applications. The number 1 of the OWASP top 10 is called “Broken Access Control”, as stated by OWASP: “Access control enforces policy such that users cannot act outside of their intended permissions. Failures typically lead to unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user's limits. ”

Oauth2 allows defining scopes that represent certain permissions. Developers should be careful in designing and implementing scopes to ensure that the access is granted only to the necessary resources.

Besides this OAuth2 uses token validation which is very important. Applications should verify the integrety and authenticity of the token they received, ensuring that they have not been altered with and that they are issued by a trusted authorization server. OAuth2 also supports token expiration, reducing the window of opportunity for attackers. Implementing short lived tokens ensures that even if a token is compromised, the potential damage is mitigated. If needed, tokens can be revoked if a user revokes access or suspicious activity is detected, this is to prevent further unauthorized access.

Because that these tokens are so valuable for attackers, they have to be stored and transferred securely that will prevent unauthorized access. When transferring they should use HTTPS as a secure channel to transmit the tokens between client and the resource server.

# Summary

Lets summarize the answers I’ve found for my sub-questions

## What is OAuth?

OAuth (Open Authorization) is a framework that makes logging in easy and secure. It allows a website or an application to access resources (like profile pictures or your name) hosted by other web applications on behalf of the user, it is the “Login with Google” or “Login with Facebook” button you see on certain websites or apps, without having to give credentials, like your password, to the original application.

## How did OAuth start?

In 2006, there was a need for authentication delegation, but a group of developers concluded that OpenID did not provide this availability. So in 2007, the group started formulating a proposal for an open protocol. Eventually, DeWitt Clinton from Google expressed his interest in endorsing the effort as a stakeholder. In the same year the group released the final draft of OAuth Core 1.0.

## What is the difference between Authentication and Authorization?

Authentication and Authorization are often used interchangeable, but fundamentally they are completely different things.

While Authentication is the process of verifying who a user is. Meanwhile Authorization is the process of verifying what the user is allowed access to.

## How does OAuth2 work?

In simple terms. When you want to login to an application, and you want to login with Google, you login with Google, then Google gives a key to the application. The application can use this key to open up certain locks with information, like a profile picture or your name. The key doesn’t open the lock to your password or your credit card information.

# Conclusion

So lets conclude this report by finally answering the main research question: **“What is the function of OAuth2?”**

The function of OAuth2 is so that you can login to an application via Google, Facebook, etc. and the application to be able to use certain data from those accounts, without ever having to give your password to the application. This also secures the login to only get access to certain data to prevent data to be retrieved by people that don’t have access to it.

# Sources

OAuth.net:  
<https://oauth.net>

Auth0 | Intro to OAuth2:  
<https://auth0.com/intro-to-iam/what-is-oauth-2>

CSO Online | What is OAuth?  
<https://www.csoonline.com/article/562635/what-is-oauth-how-the-open-authorization-framework-works.html>

Intersystems Learning Services | OAuth 2.0: An Overview  
<https://www.youtube.com/watch?v=CPbvxxslDTU&t=203s>

Auth0 | Authentication and Authorization  
<https://auth0.com/docs/get-started/identity-fundamentals/authentication-and-authorization>

StackOverflow | Difference between OpenID and OAuth  
<https://stackoverflow.com/questions/4230821/if-openid-is-dead-what-is-out-there-to-take-its-place/4230970#4230970>

Wired | OpenID: The Web’s Most Successful Failure  
<https://www.wired.com/2011/01/openid-the-webs-most-successful-failure/>

OpenID  
<https://openid.net>

OWASP

<https://owasp.org/Top10/>