Mastering OAuth 2.0

OAuth 2.0 is a powerful authorization framework that enables your application to interact with the world’s most popular service providers, allowing you to leverage their world-class technologies in your own application.

At the start of this book, you will learn what OAuth 2.0 is, how it works, and the steps involved in creating an OAuth 2.0 application. You will first look at the process of registering your client application. Following this, you will explore the different supported workflows, their differences, and when they should be used. Once this groundwork is laid, you will put that knowledge to use by creating a sample application that talks to Facebook using OAuth 2.0.

By the end of the book, you will know how to make a secure and robust OAuth 2.0 application. You will explore different tools and effective methods for debugging. In addition, you will look at important security considerations for your application. Finally, you will understand the different implications of making a native mobile application versus a traditional web application.

Who this book is written for
If you are an application developer, software architect, security engineer, or even casual programmer looking to leverage the power of OAuth 2.0, this book is for you. A basic knowledge of programming and OAuth 2.0 is recommended.

What you will learn from this book
- Discover the power and prevalence of OAuth 2.0 and use it to improve your application’s capabilities
- Step through the process of creating a real-world application that interacts with Facebook using OAuth 2.0
- Examine the various workflows described by the specification, looking at what they are and when to use them
- Get to know about the many security considerations involved in creating an application that interacts with other service providers
- Develop your debugging skills with dedicated pages for tooling and troubleshooting
- Expand your knowledge of OAuth 2.0 by discovering the many ways OAuth 2.0 can be extended
- Build your own rich, powerful applications by leveraging world-class technologies from companies around the world

Create powerful applications to interact with popular service providers such as Facebook, Google, Twitter, and more by leveraging the OAuth 2.0 Authorization Framework.
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 'Why Should I Care About OAuth 2.0?'
- A synopsis of the book’s content
- More information on Mastering OAuth 2.0
About the Author

Charles Bihis is a scientist and engineer from Vancouver, Canada. Earning his degree in computer science from the University of British Columbia, specializing in software engineering, he enjoys exploring the boundaries of technology. He believes that technology is the key to enriching the lives of everyone around us and strives to solve problems people face every day. Reach out to him on his website, www.whoischarles.com, and let's solve the world's problems together!
Preface

The Internet is a thriving and dynamic ecosystem. Living and playing within this ecosystem are many world-class services, all offering world-class technologies. Think about the massive social graph that Facebook hosts, the most up-to-date mapping system proudly owned and operated by Google, or the ever-growing professional network that is available from LinkedIn. All of these companies, and more, are presenting their world-class technologies for the world to use!

Until recently, it was very difficult to access these technologies in your own applications. Each company would create their own protocols for how to access and leverage their respective technologies. You may have heard of Yahoo!’s BBAuth, or Google’s AuthSub. These are just a couple of examples of proprietary protocols created to allow people to leverage these company’s services. Unfortunately, the trend of creating and using proprietary protocols just doesn't scale. Enter OAuth 2.0.

OAuth 2.0 is an open protocol for delegating authorization to such services, and it has become the standard authorization protocol used by companies around the world. It allows developers like you and I to access these world-class technologies and use them in our own applications! It is a fascinating problem space with an equally fascinating and elegant solution.

I've been lucky enough to work in the Identity space for the past 7 years, and during this time, I've been able to witness the evolution and progression of this protocol. Mastering OAuth 2.0 is an attempt at distilling the most important parts of the protocol, including design and usage. With a hard focus on practicality and security, this book focuses on the parts of integration that will give application developers like you and I the most benefit and mileage.

As OAuth 2.0 continues to gain adoption, and more and more services become available for developers to integrate with and leverage, I'm hoping that this book will allow you to be able to comfortably dive in and start building the next generation of world-class applications and technologies!
What this book covers

Chapter 1, Why Should I Care About OAuth 2.0?, introduces the OAuth 2.0 protocol, and discusses its purpose, prevalence, and importance.

Chapter 2, A Bird's Eye View of OAuth 2.0, takes a high-level look at the OAuth 2.0 protocol and the different workflows it describes.

Chapter 3, Four Easy Steps, enumerates the simple steps necessary to integrate with a service provider using the OAuth 2.0 protocol.

Chapter 4, Register Your Application, details the first of these four steps which covers registering your application with the service provider.

Chapter 5, Get an Access Token with the Client-Side Flow, discusses the complicated topic of gaining access to a protected resource from what we call an untrusted client.

Chapter 6, Get an Access Token with the Server-Side Flow, discusses the complicated topic of gaining access to a protected resource from what we call a trusted client.

Chapter 7, Use Your Access Token, outlines the process for exercising access to a resource once it has been granted to you.

Chapter 8, Refresh Your Access Token, talks about the process of refreshing your access once it expires.

Chapter 9, Security Considerations, discusses the many important security considerations to be made in your application. This is an important topic for any application, but is especially important given the power that this protocol allows.

Chapter 10, What About Mobile?, is a chapter dedicated to the topic of mobile devices, including phones and tablets, and all of the considerations that come with it.

Chapter 11, Tooling and Troubleshooting, talks about how to troubleshoot issues with your integration as well as how to appropriately handle errors so as to minimize user interaction.

Chapter 12, Extensions to OAuth 2.0, looks at the various ways OAuth 2.0 can be extended to satisfy a multitude of use cases.

Appendix A, Resource Owner Password Credentials Grant, takes a look at one of the supplemental supported flows in the book.

Appendix B, Client Credentials Grant, takes a look at another of the supplemental supported flows in the book.

Appendix C, Reference Specifications, enumerates the various open specifications that are referenced throughout the book.
Why Should I Care About OAuth 2.0?

As an application developer, you may have heard the term OAuth 2.0 thrown around a lot. OAuth 2.0 has gained wide adoption by web service and software companies around the world, and is integral to the way these companies interact and share information. But what exactly is it? In a nutshell…

OAuth 2.0 is a protocol that allows distinct parties to share information and resources in a secure and reliable manner.

This is the major tenet of the OAuth 2.0 protocol, which we will spend the rest of the book learning about and utilizing. Also, in this chapter, we will introduce the sample application that we will be building throughout this book, *The World's Most Interesting Infographic Generator*.

What about OAuth 1.0?

Built with the same motivation, OAuth 1.0 was designed and ratified in 2007. However, it was criticized for being overly complex and also had issues with imprecise specifications, which led to insecure implementations. All of these issues contributed to poor adoption for OAuth 1.0, and eventually led to the design and creation of OAuth 2.0. OAuth 2.0 is the successor to OAuth 1.0.

It is also important to note that OAuth 2.0 is not backwards compatible with OAuth 1.0, and so OAuth 2.0 applications cannot integrate with OAuth 1.0 service providers.
Authentication versus authorization

Before we dive into our discussion of OAuth 2.0, it is important to first define some terms. There are two terms in particular that are pivotal to our understanding of OAuth 2.0 and its uses: **authentication** and **authorization**. These terms are often conflated and sometimes interchanged, but they actually represent two distinct concepts, and their distinction is important to understand before continuing our discussion of OAuth 2.0.

**Authentication**

**Authentication** is the process of validating whether a person (or system) is actually who they say they are.

An example of this is when you go to the bank to withdraw money, and you provide your bank card and PIN to the teller. In some cases, the teller may ask for additional identification, such as your driver's license, to verify your identity. You may recognize this in other instances when you provide your username and password to a website, say, to view a document.

**Authorization**

**Authorization** is the process of determining what actions you are allowed to perform once you have been authenticated.

Referring to our previous bank example, once the teller has verified who you are, they can then proceed to fulfill your request to withdraw money. In order to do this, they must check whether you are allowed to withdraw money from the account that you are requesting (that is, you are actually the owner of the account). Relating to our website example, once you have authenticated by providing your username and password, the website will then check to see whether you are indeed allowed to see the document that you are requesting. This is usually done by looking up your permissions in some access control list.

Now that we have established the distinction between these two important concepts, we can look at what OAuth 2.0 actually is and the problems it solves.
What problems does it solve?

Have you ever logged into a site using your Google account? Have you ever posted to Pinterest and Instagram at the same time? Have you ever shared a link to your wall from any application other than Facebook? These are all examples of OAuth 2.0 in use!

At a high level, the OAuth 2.0 protocol allows two parties to exchange information securely and reliably. In more practical terms, you'll find that the most common uses of OAuth 2.0 involve two things:

- Allowing a user to log into an application with another account. For example, Pinterest allowing users to log in with their Twitter accounts. This is known as federated identity.
- Allowing one service to access resources on another service on behalf of the user. For example, Adobe accessing your Facebook photos on your behalf. This is known as delegated authority.

Both of these combine to allow the creation of powerful applications that can all integrate with each other.

Both of the scenarios mentioned in the preceding list are actually really the same scenario. In both, the user is accessing a protected resource on behalf of another party. In the first example, the protected resource is the user's account information, while in the second example the protected resource is the user's Facebook photos. This will become clearer as we explore the details of how the OAuth 2.0 protocol handles these situations.

**Federated identity**

Federated identity is an important concept in identity management. It refers to the concept that allows one service provider to allow authentication of a user using their identity with another service provider. For instance, imagine a user that logs into Foursquare and Amazon with their Facebook credentials. In this example, the user only needs to maintain a single user account, their Facebook account, which gives them access to several service providers; in this case, Facebook itself, plus Foursquare, and Amazon. They don't need to create individual accounts on Foursquare or Amazon, and therefore, don't need to maintain three separate passwords. In this sense, the user's identities across these sites are federated, as in, they are made to act as one.
Why Should I Care About OAuth 2.0?

The OAuth 2.0 Authorization Framework
Strictly speaking, the OAuth 2.0 protocol is actually an authorization protocol and not an authentication protocol. Because of this, OAuth 2.0 alone cannot provide federated identity. However, when used in a certain way, and in conjunction with other protocols, OAuth 2.0 can provide federated authentication, which is a key component to federated identity systems.

See the OpenID Connect section in Chapter 12, Extensions to OAuth 2.0, to see how the OAuth 2.0 protocol can be combined with OpenID to provide an authentication layer on top of the authorization framework described by the OAuth 2.0 specification.

Delegated authority
Delegated authority is another important concept in the identity space. It refers to the ability for a service or application to gain access to a user's resources on their behalf. Take, for instance, LinkedIn, which can suggest contacts for you to add by looking at your Google contact list. In this example, LinkedIn will be able to view your Google contact list on your behalf. Permission to access your Google contacts has been delegated to LinkedIn.

Real-life examples of OAuth 2.0 in action
Now that we understand the basic principles of OAuth 2.0, let's take a look at some everyday, real-life examples of OAuth 2.0 in action:

- StackOverflow allowing you to log in with your Google account
- Posting a status update from your phone using the Facebook mobile application
- LinkedIn suggesting contacts for you to add by looking at your Google contacts
- Pinterest allowing you to pin something from a WordPress blog
- Sharing an article to your Facebook feed from the article itself

As you can see, if you've ever done any of these things, or anything similar, you have probably already used OAuth 2.0.
How does OAuth 2.0 actually solve the problem?

In order to see how OAuth 2.0 solves this problem of sharing resources, let's look at how this problem was solved before OAuth 2.0 was created.

Without OAuth 2.0 – GoodApp wants to suggest contacts by looking at your Facebook friends

Imagine that you have just signed up for the service GoodApp. As a new user, you don't have any contacts. GoodApp wants to suggest contacts for you to add by looking at your Facebook friends. If any of your Facebook friends are on GoodApp, it will suggest that you add them.

Before the creation of OAuth 2.0, this was solved in a very insecure way. GoodApp would ask you for your username and password for Facebook. GoodApp would then log into Facebook on your behalf to get your friends. This interaction can be looked at like this:
Why Should I Care About OAuth 2.0?

Here is how it works:

1. You ask GoodApp to suggest contacts to you.
2. GoodApp responds by saying, "Sure! Just give me your Facebook username and password please!"
3. You give GoodApp your username and password for your Facebook account.
4. GoodApp then logs into Facebook using your credentials, effectively impersonating you, to request your friend list.
5. Facebook happily obliges, giving GoodApp your friend list.
6. GoodApp then uses this information to tailor suggested contacts for you.

Why is this a bad idea? There are five key reasons:

• **You have given GoodApp the power to do *anything* with your account**: This is known proverbially as giving it the "keys to the city". You have essentially given GoodApp access to everything in your account, as if they were you. Now imagine it wasn't GoodApp. Instead it was NewUnknownApp. It's easy to see how this becomes very dangerous.

• **GoodApp may save your password, and may do so insecurely**: In order for GoodApp to maintain access to your account, they would need to store your credentials. The act of storing your password is an extremely bad practice and should be avoided at all times. To make things worse, different companies enforce different standards of security, some of which are shockingly low.

• **You are giving more chances for your password to get stolen**: You are sending your username and password across the Internet. The more times you do this, the more risk there is for someone to steal it.

• **You have to change your Facebook password if GoodApp ever gets hacked**: If GoodApp somehow got compromised, your Facebook credentials will also have been compromised. You would then need to change your Facebook password as a result of GoodApp getting owned.

• **There is no way to revoke access**: If GoodApp was acquired by EvilCorp and started doing things that you didn't like, the only way to revoke access would be to change your Facebook credentials.
With OAuth 2.0 – GoodApp wants to suggest contacts by looking at your Facebook friends

Now, let's take a look at that interaction, but this time utilizing the OAuth 2.0 protocol. In this scenario, GoodApp would "ask" Facebook for your friend list. You give permission to this by logging into Facebook and approving the request. Once the request is approved, GoodApp would then be able to fetch your friend list from Facebook on your behalf.
Let's have a look at the flow:

1. You ask GoodApp to suggest contacts to you.
2. GoodApp says, "Sure! But you'll have to authorize me first. Go here..."
3. GoodApp sends you to Facebook to log in and authorize GoodApp.
4. Facebook asks you directly for authorization to see if GoodApp can access your friend list on your behalf.
5. You say "yes".
6. Facebook happily obliges, giving GoodApp your friend list. GoodApp then uses this information to tailor suggested contacts for you.

Why is this better? Five key reasons to contrast the five points in the previous example:

• **You aren't giving it the "keys to the city" anymore:** Notice, in this example, you aren't giving your Facebook username and password to GoodApp. Instead, you are giving it directly to Facebook. Now, GoodApp doesn't have to even worry about your Facebook credentials.

• **Since you aren't giving your credentials, GoodApp no longer needs to store them:** With your authority delegated from Facebook, you don't need to worry that GoodApp is storing, or even seeing, your Facebook password.

• **You send your password across the Internet less frequently:** If you already had an active session with Facebook, you actually wouldn't need to reauthenticate with them. If GoodApp has federated identities with Facebook, you would have to send your password even less frequently.

• **You don't have to change your Facebook password if GoodApp ever gets hacked:** This is because of the next point.

• **There is a way to revoke access:** OAuth 2.0 provides the ability for a service provider to revoke access to a client. If GoodApp ever got compromised, or got acquired by Evil Corp, you could go to Facebook and revoke GoodApp's access.

### Who uses OAuth 2.0?

In the previous section, we mentioned that OAuth 2.0 has become one of the most important protocols for applications and service providers today. But how important is it? Here is a short, non-exhaustive list of some of the biggest supporters of the OAuth 2.0 protocol, along with some of the capabilities that they provide:

• **Google:** You can leverage a multitude of Google's services by interacting with their APIs via OAuth 2.0
• **Facebook**: Facebook’s social graph is accessed via OAuth 2.0 and allows users to do a tremendous amount of things, including posting to their wall and sending messages

• **Instagram**: Instagram allows you to access a user’s feed and post comments and likes

• **LinkedIn**: Post comments, share links, and gather engagement statistics via the LinkedIn APIs

• **Spotify**: Query Spotify’s massive music catalog and manage user’s playlists using Spotify’s APIs

• **Foursquare**: The Foursquare API lets you look up users and places from all over the world

There are many, many more companies that use and support the OAuth 2.0 protocol. This gives developers an enormous amount of power to create amazing applications that can leverage all of these world-class services.

**Introducing "The World's Most Interesting Infographic Generator"**

The best way to learn is simply by doing it. So, to learn the concepts of OAuth 2.0, we will be building an application throughout this book that will integrate with Facebook. It will be called *The World’s Most Interesting Infographic Generator*. It will allow a user to log in with their Facebook account, request their profile data and a list of their most recent posts, and return interesting statistics about their posting habits. You can see a working example of this application at [www.worldsmostinterestinginfographic.com](http://www.worldsmostinterestinginfographic.com), or [www.wmiig.com](http://www.wmiig.com) for short.

**Summary**

In this chapter, we took an introductory look at what OAuth 2.0 is and how it is used all around us. We discussed the benefits that this protocol gives us and even looked at the kind of adoption that has taken place in the industry. It has become one of the most, if not the most, used and adopted authorization protocols on the Internet due, in large part, to the power that it gives application developers, start-ups, and corporations alike, to share information.

In the next chapter, we will look at how OAuth 2.0 provides these benefits by looking at how OAuth 2.0 actually works under the hood. Get ready!
Where to buy this book

You can buy Mastering OAuth 2.0 from the Packt Publishing website.
Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.
Click here for ordering and shipping details.