OAuth - Open Authorization

(https://github.com/learn-co-curriculum/dsc-oauth-open-authorization) (https://github.com/learn-co-curriculum/dsc-oauth-open-authorization/issues/new/choose)

Introduction

Perhaps the most common thread among all APIs available on the web is the need to ***authenticate*** yourself to prove you are who you say are. This is a major security point for web APIs, and it's no surprise that a standard protocol has evolved to help make authenticating users simple and safe. This authentication protocol is called ***OAuth***, and it provides a simple, easy integration for multi-user applications, such as web APIs, so that they can add authentication capabilities to their API in a simple, streamlined way. From our side, this makes life easy because it standardizes the authentication process so that we don't have to worry about many different authentication protocols for every different service. In this lesson, we'll look into OAuth, learn a little about how it works, and explore the benefits OAuth offers.

Objectives

You will be able to:

- Explain the role OAuth plays when working with a 3rd party API

What is OAuth?

> **OAuth stands for Open Authorization.**

OAuth is an open-source protocol created to allow the creators of APIs and other online services to easily let them share private data or assets with users. One of the biggest challenges of building multi-user applications is making sure that you only give people access to the data and functionality they're supposed to have. OAuth provides a framework for allowing authenticated access, but without the risk of having to share the original login credentials such as a password. The OAuth protocol was created in 2010 and was the brainchild of major tech companies such as Google and Twitter. It's now the most popular open standard for user authentication, is used by almost all of the major players in the tech world, such as Netflix, Amazon, Facebook, and more!

The Steps of OAuth

Prior to using OAuth, we must also register our application with the authorizer and get our c' ? Help 'se during the process. We need to set up some information about the application, like the app's name or website, and most importantly, **a redirect URI**. The authorizer later uses this to contact the requesting app and tell them that the user said yes.

> A URI (Uniform Resource Identifier) is a string that refers to a resource. The most common are URLs, which identify the resource by giving its location on the Web.

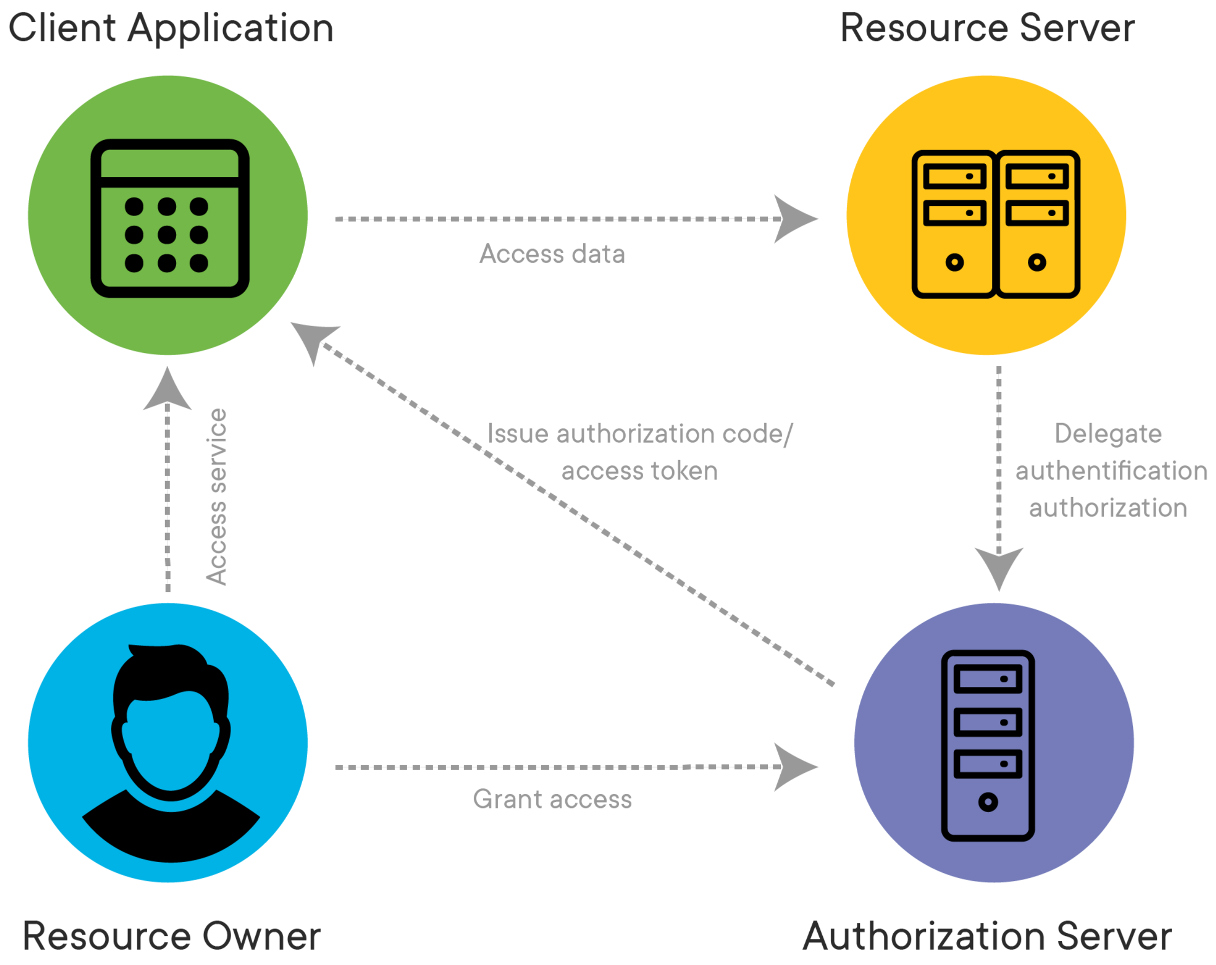
After registration, The first step is the **authorization**. Here, we send our users to the authorization server to ask for some permissions with our scope (permissions) that we would like to have. The user can see everything being requested on his behalf and confirm that they would like to grant our application access for those permissions.

The second step is the **redirect**. Redirect URIs are a critical part of the OAuth flow. After a user successfully authorizes their application, the authorization server then redirects the user back to the app with an **authorization code** in the URL. Because the redirect URL will contain sensitive information, it is critical that the service doesn't redirect the user to arbitrary locations. The authorization code is used by our application in the final act of getting the access token.

The final step is **acquisition**. This is where we finally receive our **access token** from the service provider so we can process API requests for our user. We use the authorization code we received in the redirect to our redirect url and our own application secret (which is acquired during initial registration) in order to get our user's access token. The access token can then be used to make API calls on behalf of our user.

This architecture is summarized in the image below:

Help



The following example shows a scenario where you may want to access a user's Dropbox account for storing photo/media as a part of the service you provide.

OAuth with DropBox for Single-Sign-On (SSO)

If you've ever used your Facebook or Google account to log in to a 3rd party website or app, then you've used OAuth--OAuth is what makes this sort of ***Single-Sign-On*** or ***SSO*** ability possible. To get a feel for how OAuth makes this possible, let's look at a real-world example--giving Dropbox access to the photos app on your phone or computer!

Registering with OAuth and Using SDKs

Before an application can be used with OAuth, that application must first be *registered* with OAuth. This is a process handled by the team creating the Application itself, and not something we need to worry about as users. For instance, in order for the Photo Gallery app on our Macbook to have access to Dropbox, the team behind this software at Apple will have taken the time to register their app, provide the name of it, set the permissions it will need, and other things like that. The good news is that in most cases, we don't need to write the actual code that allows our application to interface with the service we're connecting to--major companies usually create ***Software***
D Help ***its*** or ***SDKs*** that we can drop into our application and use as a library to add

functionality like SSO with our Facebook account or the ability to access files in our Dropbox account.

The OAuth Process

In order for our app to access our users' Dropbox accounts, the following steps need to happen:

1. A user must authenticate with Dropbox, verifying their identity.
2. The user must explicitly give app permission to access their Dropbox (you've probably seen these sorts of popups before when connecting apps to one another on your phone).

One quick technical note: Dropbox (and most other online services) actually use **OAuth 2** specifications. Details on OAuth2 can be viewed at the official website (https://oauth.net/2/) . In practice, most people just call it OAuth for short, but there is technically a difference between OAuth and OAuth 2 in terms of how they work and what they can do.

Once the user has completed signing in and verifying permissions, the OAuth process returns an ***Access Token*** to our application.

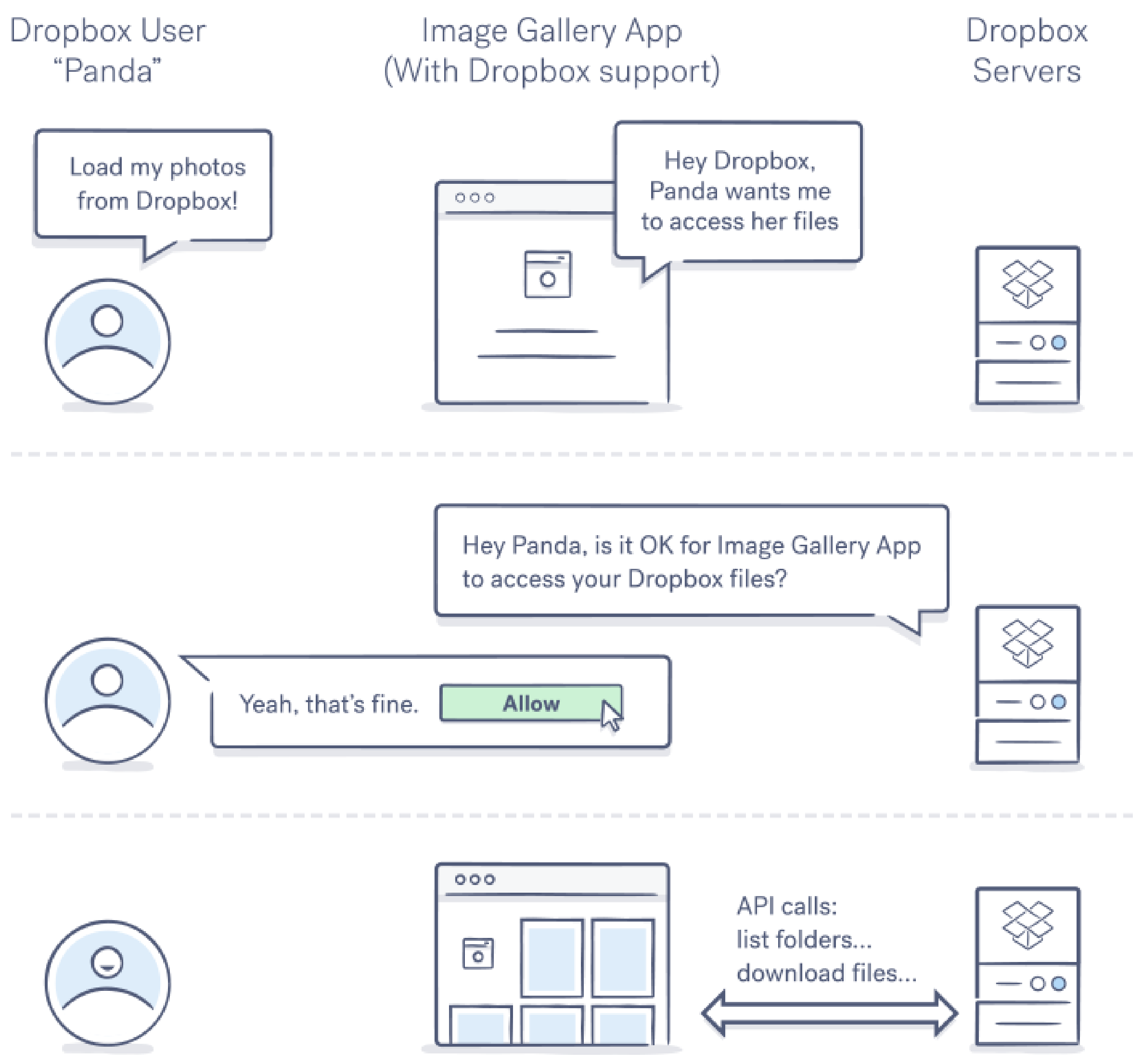
> **An access token is a special string generated by authorizer that we need to send with each subsequent API request to uniquely identify both the app and the user.**

You can think of an access token as a secret code that identifies our HTTP requests as coming from the actual user that has signed in and authenticated themselves.

For the remainder of this lesson, we'll borrow some example graphics and explanations from Dropbox's amazing OAuth Guide (https://www.dropbox.com/developers/reference/oauth-guide) , which provides a detailed explanation of exactly how OAuth works with Dropbox. We're only interested in the broad strokes of this as an example of how it all works, but if you have time, take a look at the full guide--it's quite informative and very easy to understand!

Have a look at an example scenario for an Image/gallery app that wants to access its users' Dropbox accounts for accessing or storing new images.

Help


Dropbox User "Panda"
Image Gallery App (With Dropbox support)
Dropbox Servers
Load my photos from Dropbox!
Hey Dropbox, Panda wants me to access her files
Hey Panda, is it OK for Image Gallery App to access your Dropbox files?
Yeah, that's fine.
Allow
API calls:
list folders...
download files...

The key benefit of this approach is that our app doesn't need to store or transmit the user's Dropbox password. The user will be redirected to Dropbox to authorize our app to access their Dropbox data. After the user has approved our app, they'll be sent back to the app with an authorization code. At this point, our app will exchange the authorization code for an access token which can be used to make subsequent requests to the Dropbox API for downloading/uploading contents, etc. OAuth also allows the user to authorize only a limited set of permissions and the user may choose to stop access at any time. This makes OAuth a safer and more secure form of API authorization for end users.

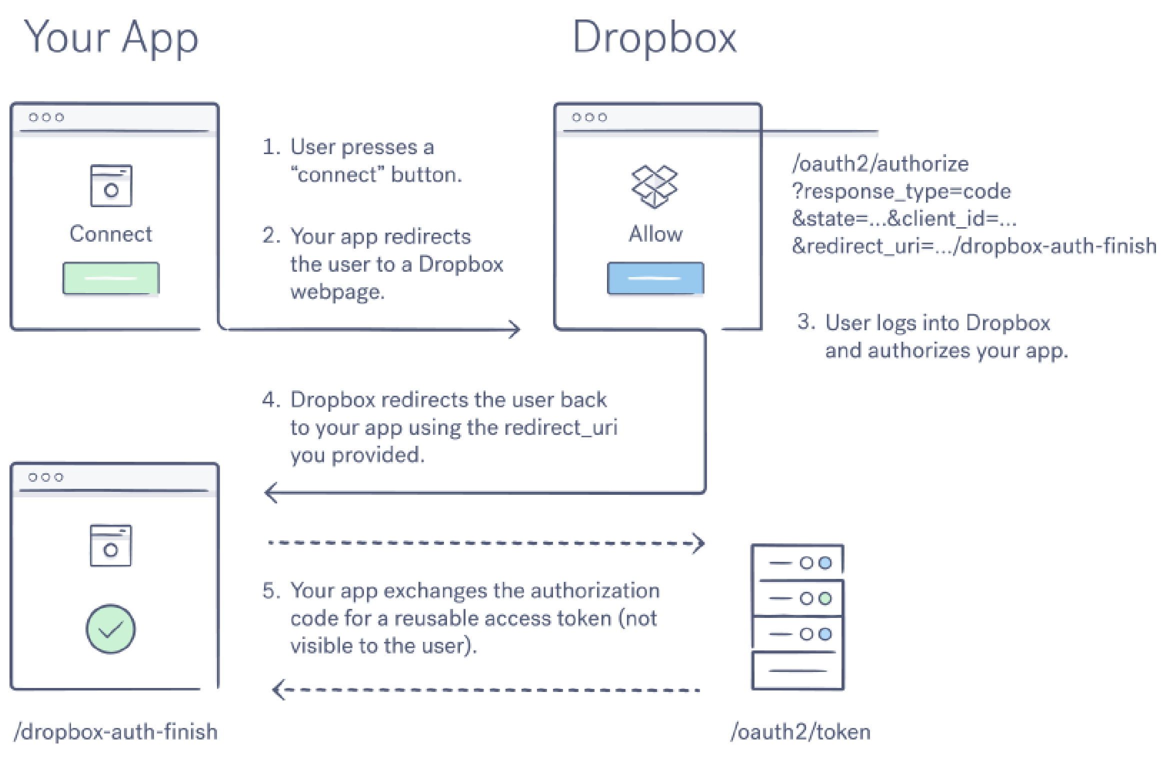
Dropbox access for web apps

If you've used SSO features before or connected apps to other apps on your smartphone, then you're probably familiar with the overall workflow of how OAuth takes users through the process of authenticating.

Help

1. The application redirects the user to a page to sign in to the account they're using, such as redirecting to the Facebook login page when you click "Login with Facebook" in a 3rd party app or game.
2. The User signs in, and explicitly provides permissions to the application or game to connect to their account--in our example, this would mean giving the Photo Gallery app on our Macbook explicit permission to access our Dropbox account.
3. The user is given an access code and redirected back to the original application. The application then signs in, and exchanges the access code for a more permanent ***Access Token*** that it can store for all subsequent connections to the service that it has authenticated with--this is the reason why our 3rd party apps can stay signed in and connected with things like our Facebook or Dropbox accounts, so that we don't have to reauthenticate every time we sign into the app!

Note that this request to exchange the authorization code for an access token takes place behind-the-scenes with a call to the /token API endpoint and is not visible to your end users. This setup is shown in the image below, which comes from the Dropbox OAuth Guide linked above.


Your App
Dropbox
Connect
1. User presses a "connect" button.
2. Your app redirects the user to a Dropbox webpage.
Allow
/oauth2/authorize ?response_type=code &state=...&client_id=... &redirect_uri=.../dropbox-auth-finish
3. User logs into Dropbox and authorizes your app.
4. Dropbox redirects the user back to your app using the redirect_uri you provided.
5. Your app exchanges the authorization code for a reusable access token (not visible to the user).
/dropbox-auth-finish
/oauth2/token

The Benefits of Using OAuth

Now that we understand how OAuth works, let's think about why it's useful. Before OAuth, authentication meant ***Direct Authentication*** through an HTTP request, where the user is prompted for a username and password. This sort of Basic Authentication is still used in plenty of places, as a p ? Help API authentication for server-side applications: instead of sending a username and pa e server with each request, the user sends an API key ID and secret. Before OAuth, sites prompted users to enter their username and password directly into a form and sites would log into the user account.

What makes OAuth so special and so effective is that it is a ***Delegated*** authorization framework for RESTful APIs. This means that it allows apps to obtain limited access (scopes) to a user's data without giving away a user's password--think about every HTTP request an app makes when it needs information from a connected Facebook account. It would be a major security risk if each of these HTTP requests still had to contain a username and password to verify that the HTTP request should have access to a certain account. By using an OAuth access token, it allows us to decouple authentication (proving we are who we say we are) and authorization (getting permissions for all we need to do).

Further reading

- OAuth 2.0 (https://oauth.net/2/)
- How to dance the OAuth: a step-by-step lesson (https://medium.freecodecamp.org/how-to-dance-the-oauth-a-step-by-step-lesson-fd2364d89742)
- OAuth, wherefore art thou? (https://medium.com/square-corner-blog/oauth-wherefore-art-thou-b7034098a0fd)

Summary

In this lesson, we looked at the OAuth process for gaining access to user-owned resources. We saw how we authorize our apps to access resources with a high level of confidentiality and security that OAuth offers. We looked at an example of how this might work with the Dropbox API and also some extra reading to see more examples of this process. Next, we'll put this into practice by seeing how this process works with the Yelp API.

How do you feel about this lesson?

Have specific feedback?

Te ttps://github.com/learn-co-curriculum/dsc-oauth-open-authorization/issues/new/choose)

Help