As part of learning backend development, I learned about OAuth and Auth0. I had heard both these terms being used interchangeably in my team, and I wanted more clarity. This article summarizes my learnings for the same.

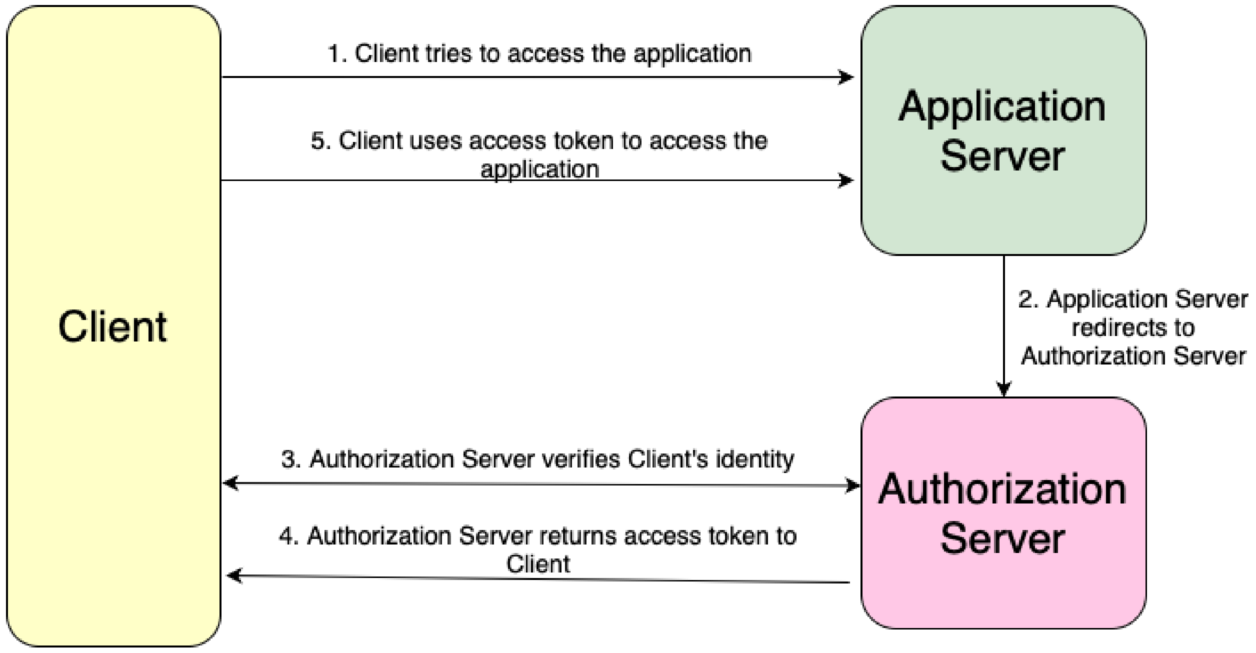
What is OAuth? OAuth stands for Open Authorization, which is an authorization framework for applications that decouples authentication from authorization. It is an open standard for authorization where we use access tokens instead of user credentials.

OAuth has two versions, OAuth 1.0a and OAuth 2.0. However these two are completely different from each other, and OAuth 2.0 is not backwards compatible with OAuth 1.0 a. Both versions have their advantages and disadvantages. Nowadays whenever people mention OAuth, they are most likely referring to OAuth 2.0 as it is more widely used and popular.

A basic OAuth flow can be described as:

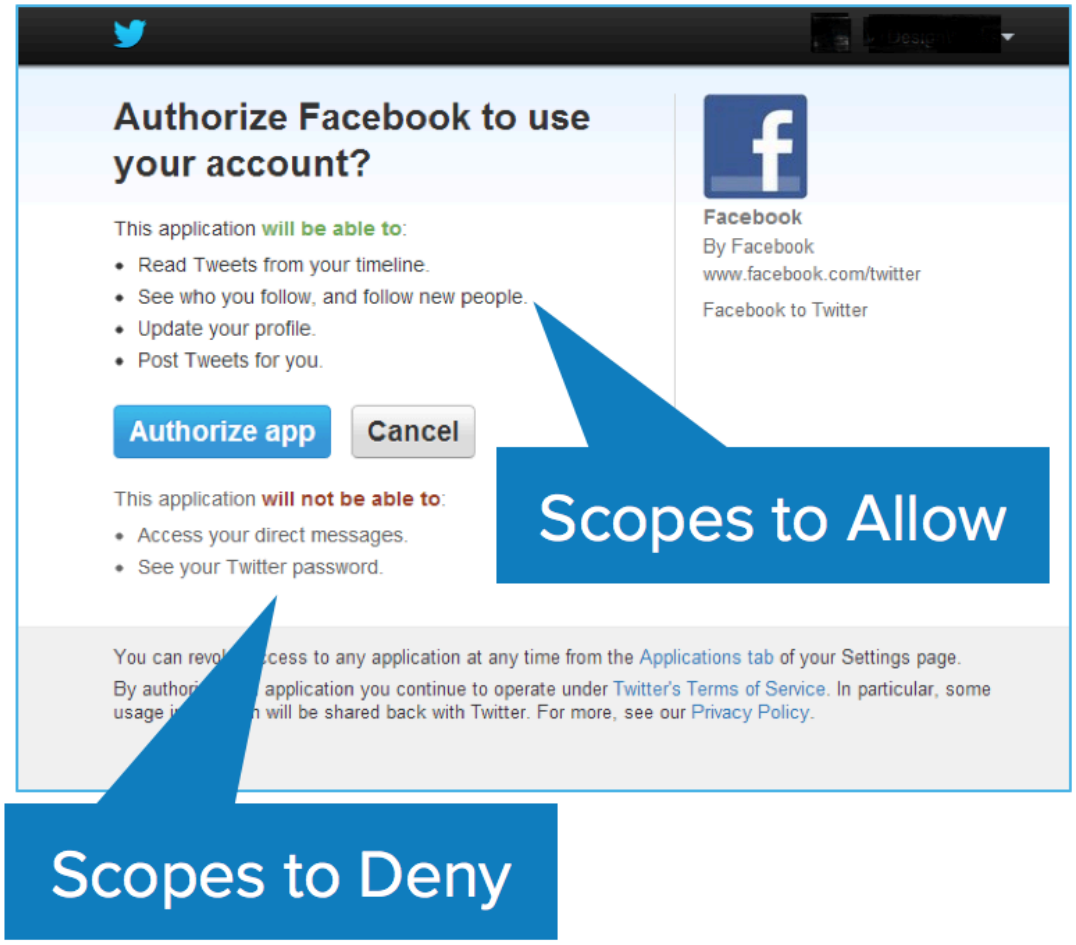
1. User access an application and tries to login
2. User gets redirected to the Authorization Server, a 3rd party that implements OAuth, which verifies the identity of the user(provided the user is already registered on that system) and then issues an access token.
3. The user uses this access token while sending the request, and the application only grants specific access to the user depending on what the user is authorized for.

The below diagram shows this flow:


Client
1. Client tries to access the application
5. Client uses access token to access the application
Application Server
2. Application Server redirects to Authorization Server
3. Authorization Server verifies Client's identity
4. Authorization Server returns access token to Client
Authorization Server

OAuth is build on the following central components:

1. Scopes and Consent - Scope is a mechanism to limit an application's access to a user's account. An application can request one or more scopes, and the user can consent to these scope requests.
   The below diagram shows a request for consent to be granted.



[redacted]

Authorize Facebook to use your account?

This application **will be able to**:

- Read Tweets from your timeline.
- See who you follow, and follow new people.
- Update your profile.
- Post Tweets for you.

Authorize app | Cancel

This application **will not be able to**:

- Access your direct messages.
- See your Twitter password.

Facebook
By Facebook
www.facebook.com/twitter
Facebook to Twitter

You can revo ccess to any application at any time from the Applications tab of your Settings page.
By authori application you continue to operate under Twitter's Terms of Service. In particular, some usage i n will be shared back with Twitter. For more, see our Privacy Policy.

2. Actors - There are 4 parties involved in this process:
   a. Resource server - the api that stores the data needed by the app
   b. Resource owner - the owner of the data in the resource server, the user who authorizes an application to access their account
   c. Client - the app that wants to access the data, but before that it needs to be authorized by user
      i. Can be public or confidential

        ii. Confidential clients are trusted to store secrets - cannot be reverse engineered to obtain secrets. Eg., smart tv, protected clients
        iii. Public clients not trusted - Eg., apps on app store
    d. Authorization Server - The main OAuth engine, that Verifies identity of user, then issues access token to app. It has two main endpoints:
        i. Authorize the end point
        ii. Get the token
3. Tokens - It is a string that the OAuth client uses to make requests to the resource server. They can be access tokens or refresh tokens.
    a. Access tokens are used by the client to gain access to the Resource Server (API). They have a short lifespan.
    b. Refresh tokens are used to obtain new access tokens when the previous ones expire. They have longer lifespans than access tokens.

The token can contain information about the user and can be configured to contain other information too. Examples include JWT(JSON Web Token) where the tokens are generated by applying an algorithm on specific data, and the data can also be decoded back from the token.

4. Flows - There are many flows that implement OAuth, but what we'll go over here is the most popular Authorisation Code Channel flow.
   This is used in server-side apps where the source code is not publicly exposed. Here an Authorization Code is exchanged for a token. The authorization code is initially provided on first time login, which is then sent along with client id and client secret to the authorization server when requesting tokens.

Auth0 is a platform that provides authentication and authorization services for applications. It implements the OAuth 2.0 protocol. Now as part of learning Auth0, I tried to implement it on a REST API created using Spring Boot and Kotlin. The setup was pretty simple and straight-forward, and these are the following steps I followed:

1. Log in to Auth0 and create an API. This involves filling out the name, identifier(url) and signing algorithm.
   This generates the credentials, and we need to note down the following parameters: identifier, domain, client secret and client ID.

2. Add necessary auth0 dependencies to the app.
   org.springframework.boot:spring-boot-starter-security
   org.springframework.security:spring-security-oauth2-resource-server
   org.springframework.security:spring-security-oauth2-jose
   org.springframework.security:spring-security-config

3. Add the configuration in the application.yml file. A sample config file is shown below:

```
None
    auth0:
      audience: { Domain }
    spring:
      security:
        oauth2:
          resourceserver:
            jwt:
              issuer-uri: https://{API Identifier}/
```

4. Add 2 security classes - Audience Validator and SecurityConfig
    a. AudienceValidator should confirm that the access token is actually the one for the API
    b. SecurityConfig should validate the access token and specify the levels of access you grant to the API endpoints
5. To get the access token - run the following command:

```
None
curl --request POST \
    --url https://{Domain}/oauth/token \
    --header 'content-type: application/json' \
    --data '{
        "client_id": "{Client ID}",
        "client_secret": "{Client Secret}",
        "audience": "{API Identifier}",
        "grant_type": "client_credentials"
    }'
```

6. Now use the token in your request as a Bearer token in the header. The header key should be “Authorization” and value should be “Bearer {token}”