OAuth in Android

What is OAuth?

- Open standard for Authorization
- "Used as a way for Internet users to authorize websites or applications to access their information on other websites but without giving them the passwords"
- "Allows notifying a **resource provider** (e.g. Facebook) that the **resource owner** (e.g. you) grants permission to a **third-party** (e.g. a Facebook Application) access to their **information** (e.g. the list of your friends)"

Different Parts to OAuth

- Client
  - Application that wants the credentials
- Provider
  - Third party like Google, Facebook, or Twitter that provides the OAuth service
- Owner
  - The person/user with the Google, Facebook, or Twitter account

Example

- Let's say you just started (yet) another social network
  - You need a way to get a list of a user's contacts
  - New users probably won't want to add all their contacts manually
  - The user also won't want to give you access to their complete contact book

Example

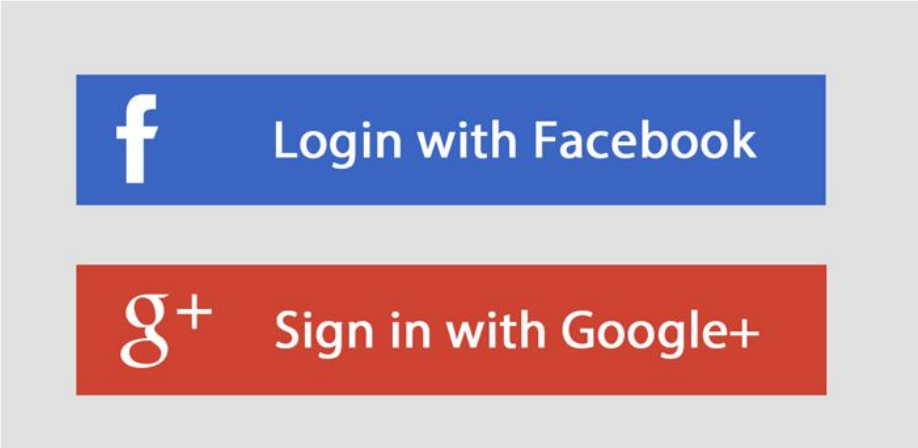
- This is where OAuth comes in
  - Instead of asking the user to manually fill in all their contacts or asking for complete access to their contact book, we can ask for partial access
  - We could use something like Gmail for access to a user's contacts
    - We can only ask for permission to READ their contacts
    - The contacts will already be there, so no overhead for the end user

Example

- OAuth allows for the ability to grant different levels of access
  - Read only for the contacts
  - The user could permit only access to name and phone number of each contact
- OAuth also allows the user to revoke access at any time
  - If they don't like our social network they can revoke GMail contact access

Common Uses

- Facebook login, Gmail login, Github login, etc....
- Custom OAuth service that you create

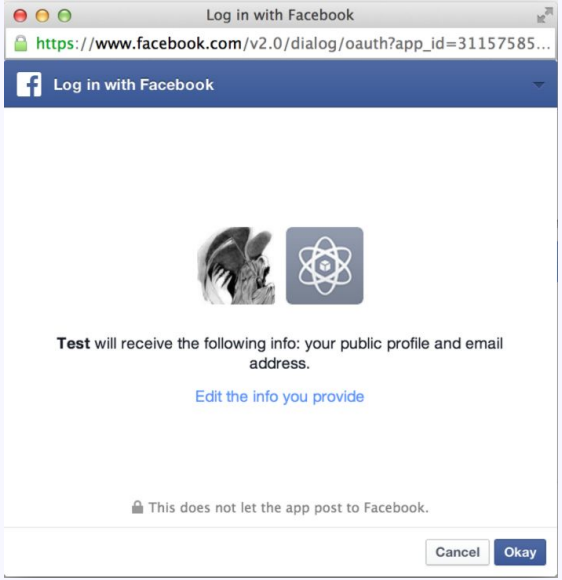


OAuth Workflow - Step 1

- User visits your website or app, and wants to login without creating a completely new account
- User clicks on a “login with Facebook” button

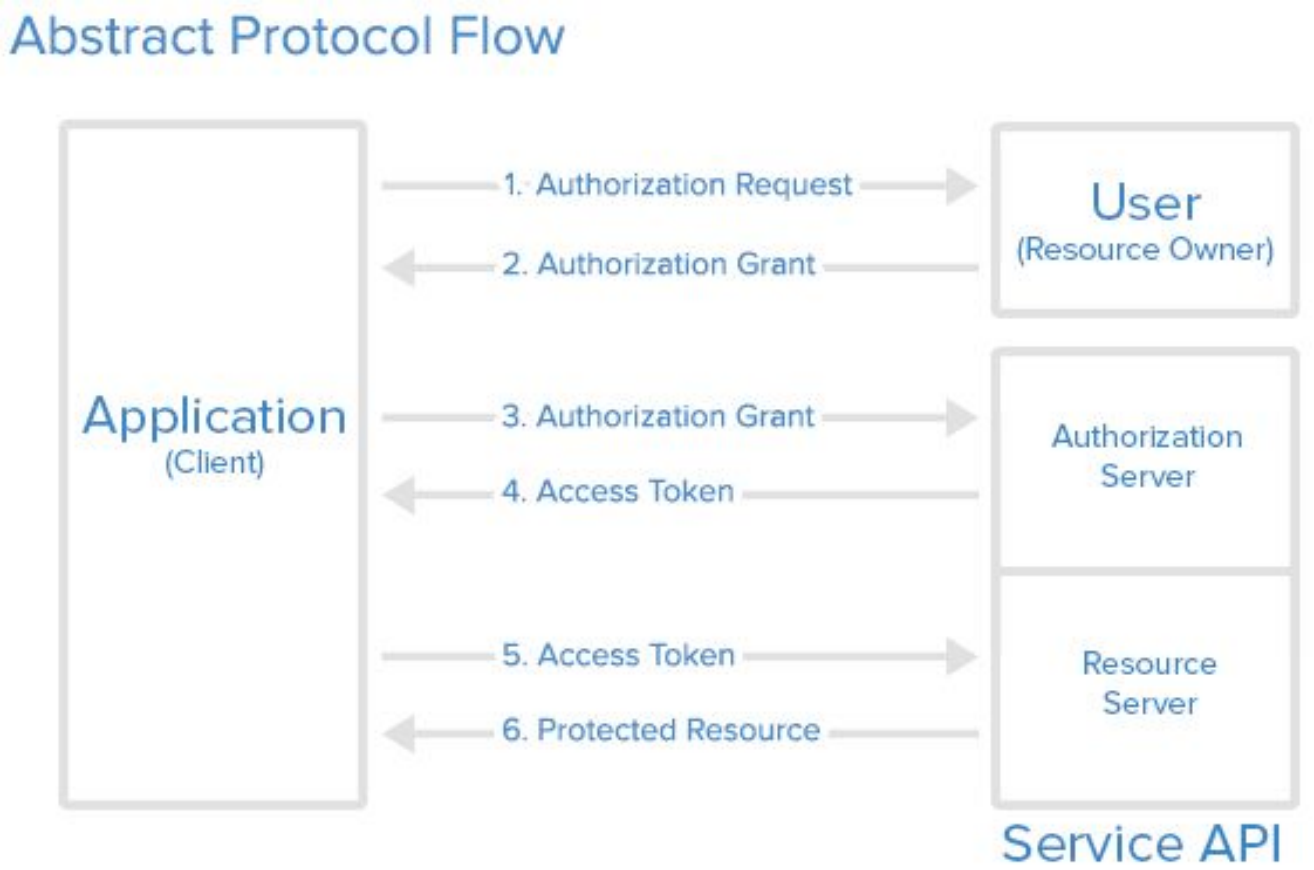
OAuth Workflow - Part 2

- Third party's login screen pops up
- User logs in and is presented with a permissions page
- If the user accepts, the page/app redirects to the original login screen



OAuth Workflow - Part 3

- The third party also returns an access token upon return to the original screen
- This access token is then used to access the user's content (contacts, email, etc.)
  - The application makes a request to OAuth backend with the token
  - If the token is verified information is returned


Abstract Protocol Flow
Application
(Client)
1. Authorization Request
2. Authorization Grant
3. Authorization Grant
4. Access Token
5. Access Token
6. Protected Resource
User
(Resource Owner)
Authorization
Server
Resource
Server
Service API

Implementing OAuth in Android

- We will be going over Google's OAuth 2.0 implementation
- I will be presenting a high level overview of the execution

Step 1

CS196

- Create a project on the Google developer console

Google APIs Project

IAM & Admin

All projects

IAM

Quotas

Service accounts

Labels

Settings

Projects

CREATE PROJECT

DELETE PROJECT

Filter by name, ID, or label

Columns

Project name	Project ID
CS196	cs196-dd1c8
CTC-Prototype	ctc-prototype
helpq-beta	helpq-beta
My Project	serene-radius-124301
My Project	fresh-span-131723
My Project	galvanized-case-131704
Sendbloom-testing	sendbloom-testing

Projects shut down and pending deletion

CS196

IAM

Labels

Add members

Select a role

Add

Search members

Filter by name or role

Owner (1 member)

Full access to all resources.

Editor (2 members)

Edit access to all resources.

Step 2

- Add Google services to your project gradle

1. Add the dependency to your project-level `build.gradle`:

```
classpath 'com.google.gms:google-services:3.0.0'
```

2. Add the plugin to your app-level `build.gradle`:

```
apply plugin: 'com.google.gms.google-services'
```

```
apply plugin: 'com.android.application'
    ...

    dependencies {
        compile 'com.google.android.gms:play-services-auth:9.8.0'
    }
```

Step 3

- Configure GoogleSignInOptions object
  - In the onCreate method, configure the object to access the data that your application needs
  - DEFAULT_SIGN_IN parameter gives access to basic profile information

```
// Configure sign-in to request the user's ID, email address, and basic
// profile. ID and basic profile are included in DEFAULT_SIGN_IN.
GoogleSignInOptions gso = new GoogleSignInOptions.Builder(GoogleSignInOptions.DEFAULT_SIGN_IN)
        .requestEmail()
        .build();
```

SignInActivity.java

Step 4

- Create a GoogleApiClient object
  - Pass in the GoogleSignInOptions object from step 3

```
// Build a GoogleApiClient with access to the Google Sign-In API and the
// options specified by gso.
mGoogleApiClient = new GoogleApiClient.Builder(this)
        .enableAutoManage(this /* FragmentActivity */, this /* OnConnectionFailedListener */)
        .addApi(Auth.GOOGLE_SIGN_IN_API, gso)
        .build();
```

SignInActivity.java

Step 5

- Add the Google Sign In button to the view
  - Use the SignInButton in XML
  - Use the SignInButton java class in your code

```
// Customize sign-in button. The sign-in button can be displayed in
// multiple sizes and color schemes. It can also be contextually
// rendered based on the requested scopes. For example. a red button may
// be displayed when Google+ scopes are requested, but a white button
// may be displayed when only basic profile is requested. Try adding the
// Scopes.PLUS_LOGIN scope to the GoogleSignInOptions to see the
// difference.
SignInButton signInButton = (SignInButton) findViewById(R.id.sign_in_button);
signInButton.setSize(SignInButton.SIZE_STANDARD);
signInButton.setScopes(gso.getScopeArray());
```

SignInActivity.java

Step 6

- Create an onClick method for the sign-in button
  - Create a signIn() helper function to call in the onClick function
  - Create a sign-in intent

```
private void signIn() {
    Intent signInIntent = Auth.GoogleSignInApi.getSignInIntent(mGoogleApiClient);
    startActivityForResult(signInIntent, RC_SIGN_IN);
}
```

SignInActivity.java

Step 7

- Create an onActivityResult function in your activity
  - Use a GoogleSignInResult object to retrieve the result from the intent

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    // Result returned from launching the Intent from GoogleSignInApi.getSignInIntent(...);
    if (requestCode == RC_SIGN_IN) {
        GoogleSignInResult result = Auth.GoogleSignInApi.getSignInResultFromIntent(data);
        handleSignInResult(result);
    }
}
```

SignInActivity.java

Step 8

```java
private void handleSignInResult(GoogleSignInResult result) {
    Log.d(TAG, "handleSignInResult:" + result.isSuccess());
    if (result.isSuccess()) {
        // Signed in successfully, show authenticated UI.
        GoogleSignInAccount acct = result.getSignInAccount();
        mStatusTextView.setText(getString(R.string.signed_in_fmt, acct.getDisplayName()));
        updateUI(true);
    } else {
        // Signed out, show unauthenticated UI.
        updateUI(false);
    }
}
```

SignInActivity.java

Links

https://developers.google.com/android/guides/http-auth

https://developers.google.com/identity/sign-in/android/sign-in