**Software Security - IE5042**

**Assignment 02**

**MS21927652 – Vithanage S A**

**How OAuth2 Works**

Native clients are treated differently by Facebook, Google, and the majority of other OAuth2 providers than web clients. Native clients are thought to be more secure, as they receive long-lasting tokens and refresh tokens. Web clients receive considerably shorter tokens, which expire when the user shuts the browser or does not click on the page for a long period.

These are the typical procedures for logging into OAuth2:

1. The user attempts to perform an action that necessitates authentication. It may be as easy as launching an app or clicking a "Log in" button to do this.
2. The app or website detects that the user has not yet logged in and initiates the login procedure. It does this by creating a web page and sending it to a particular URL provided by Facebook, Google, or whichever another website provides your OAuth2.

The opening of a new browser window for the OAuth2 provider is a crucial step. This is what allows providers to show their own log-in forms and request whatever log-in information they require from each user. In most programs, an embedded web view is utilized to do this.

You must also transmit some URL parameters that indicate the provider who you are and what you want to accomplish, in addition to the provider's log-in URL:

Client id: This identifies your app to the OAuth2 provider. To receive a client ID, you'll need to register your app ahead of time.

Redirect URI: When you're finished, tell the provider where you want to go. This might be a return to the home page for a website, or a page that closes the web view for a native app.

Response Type: This informs the supplier of what you expect in return. This value is usually either token or code, depending on whether you want an access token or a code. This value can also be used by providers to offer various sorts of data.

Scope: This informs the supplier of the data your program needs. This is how Google learns about you. Quora has asked for access to your contact management system. The scopes of each service provider are different.

The provider takes over once your app opens the web view. They may only ask for a login and password, or they could offer you with many screens asking for everything from your favorite teacher's name to your mother's maiden name. It's all up to them. The crucial aspect is that after the provider is finished, they will redirect you and present you with a token.

**OAuth2 Workflow for Facebook Login Application**

A picture containing table

Description automatically generated

Users will attempt to log into our application at first. They will do so by clicking the Login with Facebook option. The user will then be prompted to input their Facebook credentials in a dialog box. Finally, the user grants access to part of their Facebook information.

Our Angular client obtains the access token from the Facebook server after granting access. For the time being, the Angular client allows us to quickly access Facebook data.

However, the access token must be visible to the server as well. The Angular application does this by sending a request to the server along with the access token. The backend sends a verification request straight to the Facebook server to validate the token.

If the token is found to be legitimate, the Facebook server returns the user's profile information. The backend checks that the user profile data is correct and then creates a new user in the application after getting that data. The user profile will be updated instead if the user already exists in the backend.

The backend server will then generate a JWT to identify the user. The token is then returned to the client application as a response. The client app will save the token so that it may transmit it along with the request when sending requests to the server.

**Access Tokens**

An access token is a string that uniquely identifies a user, app, or Page and may be used by the app to perform graph API requests. When someone uses Facebook Login to connect with an app and authorizes the permission request, the app receives an access token that grants temporary, secure access to Facebook APIs. Access tokens can be gained in a variety of ways.

The token contains information such as when it will expire and whose app produced it. The bulk of API requests on Facebook require an access token due to privacy checks.

There are different types of access tokens to support different use cases:

1. User Access Token

Any time the app requests an API to read, edit, or write a specific person's Facebook data on their behalf, this type of access token is required. User access tokens are often collected through a login dialog, and they need a person to grant your app permission to receive one.

1. App Access Token

To alter and view app settings, you'll need this type of access token. It's also possible to utilize it to share Open Graph activities. It's produced using a pre-agreed secret between the app and Facebook, and it's then used to alter app-wide settings during calls. A server-to-server communication is used to retrieve an app access token.

1. Client Token

The client token is an identifier that you may use to identify your app in native mobile binaries or desktop programs. Because it's integrated in programs, the client token isn't supposed to be a hidden identity. The client token is only used to access a small fraction of app-level APIs. The client token may be found on the dashboard of your app. We won't go over the client token in this paper because it's rarely used. Instead, any API documentation that uses the client token will cover it.

Although each platform generates access tokens through different APIs, all platforms follow the basic strategy to get a user token:

Diagram

Description automatically generated

**How to create a Facebook App to generate App ID and App Secret**

To generate a Facebook app id, go to the Facebook Developer Dashboard and follow the procedures outlined here:

Step 1: Invoke the first step by clicking on the “Create App” button.

Graphical user interface

Description automatically generated with medium confidence

Step 2: The previous action manifests a Create an App modal. Add the app name and app contact email the click on the Create App large button to go to the next step.

Graphical user interface, text, application

Description automatically generated

Step 3: Subsequently, a list of product cards displays on the screen. Out of these innumerable options, you have to look for a Facebook Login card, click on the Set-Up link.

Graphical user interface, application, website, Teams

Description automatically generated

Step 4: On the next page, you need to choose the required platform out of iOS, Android, Web, and Other options. For this tutorial, we selected the Web option.

Graphical user interface

Description automatically generated with medium confidence

Step 5: At this point, you have to add the url of your app where you have to enable the Facebook login service. For instance, we are working with localhost, so we added the default angular development server url.

Graphical user interface, text, application, email, Teams

Description automatically generated

Step 6: At last, you have to copy your App ID; you can find it under the My Apps page within your Facebook app card.

Graphical user interface, application

Description automatically generated

Since Facebook only allows https request, we will be running the angular application using its default self-signed certificate.

Command – **ng serve –ssl**

Text

Description automatically generated

Below is the home view before logging into the application

Graphical user interface, text, application

Description automatically generated

Once the user clicks on the login with Facebook button then the user will be prompted with a pop requesting the user to enter their credentials. If the credentials are approved then the user will be able to access the application and view their contents such as profile information, meta data, and privileges.

Graphical user interface, text

Description automatically generated

Profile information is sent in the response along with the access token.

Graphical user interface, text, application, email

Description automatically generated

<https://graph.facebook.com/5590791560949893?metadata=1&access_token=EAAHdHNBwzcMBAAz3bwt6ygCsDF1i0M8RVZB3t9VfZBoYoxAzJELAxJBZCHxyWQUPEhHZBUxL9kPOwIiZBQDoeS9z4pInNaGw0FPZAiZASVrzAx9OK19Ese4B7yTHokz82ZAD5SRxAnEnZBYuudiZA4EGCB0pPCIJfTkgSl6fX22LU2dYo1k6yOldt9qa4O7CVkNBYEsN3qSVVXPVWGvNXz7qZBI>

Using the above request, we can obtain all the meta information related to the logged in user. The response also returns a list of connection URLs where we can use them to directly access the other resources using either a curl or postman request.

Background pattern

Description automatically generated

<https://graph.facebook.com/v13.0/5590791560949893/permissions?access_token=EAAHdHNBwzcMBAAz3bwt6ygCsDF1i0M8RVZB3t9VfZBoYoxAzJELAxJBZCHxyWQUPEhHZBUxL9kPOwIiZBQDoeS9z4pInNaGw0FPZAiZASVrzAx9OK19Ese4B7yTHokz82ZAD5SRxAnEnZBYuudiZA4EGCB0pPCIJfTkgSl6fX22LU2dYo1k6yOldt9qa4O7CVkNBYEsN3qSVVXPVWGvNXz7qZBI>

Using the above request we can obtain the fundamental privileges with which the user is granted initially.

Background pattern

Description automatically generated with low confidence

Below images display all the requests in the network tab which were initiated from the angular application towards the Facebook server accessing the resources using the access token.

A screenshot of a computer

Description automatically generated

Graphical user interface, application, Word

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, Word

Description automatically generated