**Software Security Assignment 2 – OAuth2**

|  |  |  |
| --- | --- | --- |
| Name: **Harindu Dharmarathne** | Student No: **MS21909764** | Subject: **SS** |

1. Introduction to OAuth2

OAuth2 is acronym for Open Authorization version 2, which is an open standard framework for authorization to offer web or application access to protected resources in a secure manner. This can be further described as providing clients a secured delegated way of accessing other servers and server resources on behalf of the resource owner without sharing access credentials with the less-trusted third party services [1].

This allows the client applications to offload the authentication tasks to separate purpose-built authentication servers or hosted services such as Google, Facebook and Microsoft which eliminates the complexity and cost of maintaining secured authentication infrastructure [2].

1. Working Principal of OAuth2

As OAuth is an authorization framework, it is engineered to provide temporary access rights to set of protected resources using “Access Tokens”; this token contains long string data to prove the requestee is legitimately allowed to access requested resources [3].

**Main Components of the OAuth2 Infrastructure**

* **Client** – This is usually a mobile app or a web that request to access protected resources on behalf of the resource owner with his consent [4].
* **Resource Owner** – This is the “end user” or the person who owns the protected resource who is willing and the absolute right to offer access rights [4].
* **Resource Server** – The service or application that contains protected resources which the access is granted only to clients with valid OAuth tokens [4].
* **Authorization Server** – The service or application that accepts requests from clients for access tokens; once the consent of the resource owner is validated, it issues OAuth access tokens [4].
* **Protected Resource** – Protected data available only for authorized clients. This data can be anything from contact list to health records to other sensitive data [4].

Following diagram shows the generic data flow between client, resource owner, authorization server and the resource server.

Diagram

Description automatically generated

Figure 1: OAuth2 Generic Data Flow [2]

1. Demo App

For demonstrating the OAuth workflow, a web application based on NodeJS has been developed which allow users to save files to Google Drive after authenticating them using Google account. This application runs on NodeJS runtime environment and followed model-view-controller (MVC) design pattern.

Following is the detail explanation of the application technologies used.

* Runtime Environment – NodeJS v10.19
* Google Drive API v3
* Google Sign-In API
* Modules Used
* Multer – For multi-part data uploading
* Request – For making HTTP calls
* ExpressJS – For building application functions to upload files into Google Drive
* IDE – JetBrains PyCharm 2020.3
* Code Revisioning – Git 2.30.1
* Testing Environment – Ubuntu 20.04.2 LTS

**Web UI of the Demo App**

Text

Description automatically generated with medium confidence

Figure 2: User Interface of Demo App

1. OAuth2 Data Flow of the Demo App

Following is the data flow of demo app and the Google authorization server to get the upload access to Resource server (Google Drive) using Authorization Token grant type. The workflow starts with the user accessing the web page and select the option of signing using Google account.

In order to use Google OAuth2 service for 3rd party integrations, it requires to create valid Google Application with subscriptions to right Google APIs. In this demo, primarily two Google APIs (Google Drive API and Google Sign-In API) have been used and subscribed to the Google application created [5].

* Google Sign-In API – This API is used to activate the OAuth2 client authentication.
* Google Drive API v3 – This is the 3rd version of Google Drive API which allows authorized clients to manipulate digital assets stored in the Google Drive.

Graphical user interface, text, application, chat or text message

Description automatically generated

Figure 3: OAuth2 Data Flow

Step 1 - Authorization Token Request

The authorization token request acquisition involves with two steps as below.

1. Client application requests the token request from the authorization server
2. Client application request access token using the authorization code.

* Following is the function to create OAuth2 URL to get the token request by calling Google OAuth endpoint located in <https://accounts.google.com/o/oauth2/v2/auth>.
* The encodeURIComponent() contains the different scopes of Google Drive API to be called for file/image related tasks such as viewing file, sending, receiving file content.

Text

Description automatically generated

Figure 4: Generate OAuth URL

* The authorization request URL is generated during the runtime, and it points to the API endpoints specified in the getOAuthURL() function.

Text

Description automatically generated

Figure 5: URL generation at Runtime

* Once the user clicks on the “login” action, the client application will be redirected to the Google login screen where user has to provide his consent.

Step 2 - User Login Prompt with Consent

In this stage, user must provide his consent to use Google account along with the user credentials (username, password) to authenticate the client app to upload digital assets to the Google Drive.

Graphical user interface, application

Description automatically generated

Figure 6: User Consent Screen

Step 3 - Authorization Code

Once the owner accepts to share his Google account to authenticate the Demo web application, the Google OAuth server will issue an authorization code. Also, user will be redirected to the pre-defined redirection URL (in this case to own server IP address) which the URL contains authorization code.

Text

Description automatically generated

Figure 7: Get Authorization Code

Step 4 - Exchange Authorization Code for Access and Refresh Token

Following function exchanges the authorization code to receive access and refresh tokens by submitting a POST request to the <https://www.googleapis.com/oauth2/v4/token>. Google OAuth service only allow the approved OAuth clients to use its services. Therefore, Google OAuth ClientID and Client Secret is kept safely and used along with the POST request.

Text

Description automatically generated

Figure 8: Exchanging Authorization Code for Access and Refresh Tokens

Step 5 - Access Token and Refresh Token

Once the user application successfully completes exchange of codes with Google OAuth server will respond with following information which will be used for file manipulation tasks such as upload, delete and view of the file contents.

* Access Token
* Token Type (the type is Bearer)
* Access Token Expiry Time (default value is 3600 seconds)
* Refresh Token
* Following code stores the access token and refresh token extracted from the response

Text

Description automatically generated

Figure 9: Storing Access & Refresh Tokens

Step 6 - API Access Token

In this step, client application accesses the resource server (Google Drive) using the API access token retrieved from the previous step. In this demo app, it is a file upload to the Google Drive. In the following function, the API access token is specified in “Authorization” with the token type of “Bearer”

Text

Description automatically generated

Figure 10: Uploading File to Google Drive

1. Reference

[1] J. Kubovy, C. Huber, M. Jäger, and J. Küng, “A Secure Token-Based Communication for Authentication and Authorization Servers,” in Future Data and Security Engineering, Springer International Publishing, 2016, pp. 237–250. doi: 10.1007/978-3-319-48057-2\_17.

[2] ”What is OAuth 2.0 and what does it do for you? – Auth0”, *auth0*, 2021. [Online]. Available: <https://auth>0.com/intro-to-iam/what-is-oauth-2/. [Accessed: 04- Jul- 2021].

[3] "Introduction to OAuth 2.0  |  Apigee Edge  |  Apigee Docs", *Apigee Docs*, 2021. [Online]. Available: https://docs.apigee.com/api-platform/security/oauth/oauth-introduction. [Accessed: 04- Jul- 2021].

[4] R. Boyd, *Getting started with OAuth 2.0*. Beijing: O'Reilly, 2012.

[5] "Using OAuth 2.0 to Access Google APIs  |  Google Identity", *Google Developers*, 2021. [Online]. Available: https://developers.google.com/identity/protocols/oauth2. [Accessed: 04- Jul- 2021].

1. Appendix

Source Code (in text format)

* Drive.api.js

// Creating API for user authentication and multipart file upload  
  
"use strict";  
  
// Load client parameters from ENV file  
require("dotenv").config();  
  
var fs = require("fs"),  
 multer = require("multer"),  
 request = require("request"),  
 express = require("express"),  
 ***router*** = express.Router();  
  
// Google OAuth client id, secret and redirect uri configured from .env  
const \_CLIENT\_ID = process.env.CLIENTID,  
 \_CLIENT\_SECRET = process.env.CLIENTSECRET,  
 \_REDIRECT\_URL = process.env.REDIRECTURL;  
  
var access\_token = "";  
  
var oauthUrl = getOAuthURL();  
  
// "Multer" is used for multi-part file uploading  
var upload = multer({  
 dest: "uploads/"  
}).single("image");  
  
// Create OAuth2 URL with Client ID, Response Type and Redirect URL  
function getOAuthURL() {  
 // authorization uri  
 var authUrl = "https://accounts.google.com/o/oauth2/v2/auth?";  
 // access\_type  
 var access\_type = "access\_type=offline&";  
 // scopes  
 var scope =  
 "scope=" +  
 encodeURIComponent(  
 "https://www.googleapis.com/auth/drive.file" + " " +  
 "https://www.googleapis.com/auth/drive.metadata.readonly" + " " +  
 "https://www.googleapis.com/auth/plus.me"  
 ) +  
 "&";  
 // response type  
 var response\_type = "response\_type=code&";  
 // client id  
 var client\_id = "client\_id=" + \_CLIENT\_ID + "&";  
 // redirect URL  
 var redirect\_uri = "redirect\_uri=" + encodeURIComponent(\_REDIRECT\_URL);  
  
 // prepare url and return  
 var \_oauthUrl = authUrl + access\_type + scope + response\_type + client\_id + redirect\_uri;  
 ***console***.info(["drive.api.js" + " : The OAuth URL is " + \_oauthUrl]);  
 return \_oauthUrl;  
}  
  
// Get the Authorization Code  
***router***.get(  
 "/auth",  
 (req, res) => {  
 ***console***.info(["drive.api.js" + " : router.get1"]);  
 res.json({ url: oauthUrl });  
 },  
 (err) => {  
 ***console***.error(err);  
 res.send(500);  
 }  
);  
  
// Get the OAuth Callback  
***router***.use("/auth/oauthcallback", (req, res) => {  
  
 ***console***.info(["drive.api.js" + " : router.use"]);  
  
 var session = req.session;  
 var code = req.query.code;  
  
 // retrieve access\_token by sending authorization code  
 var url = "https://www.googleapis.com/oauth2/v4/token";  
 request(  
 {  
 uri: url,  
 method: "POST",  
 form: {  
 code: code,  
 client\_id: \_CLIENT\_ID,  
 client\_secret: \_CLIENT\_SECRET,  
 grant\_type: "authorization\_code",  
 redirect\_uri: \_REDIRECT\_URL  
 },  
 headers: {  
 "Content-Type": "application/x-www-form-urlencoded"  
 }  
 },  
 (err, response, body) => {  
 if (err) {  
 return ***console***.error(err);  
 }  
  
 // set the access\_token  
 var json = ***JSON***.parse(body);  
 access\_token = json.access\_token;  
 session["tokens"] = body;  
  
 // redirect to home page for upload  
 res.redirect("/home");  
 }  
 );  
});  
  
// File Upload  
***router***.post("/upload", upload, (req, res) => {  
 var file = req.file;  
  
 // Construct the upload request based on the Google Drive v3  
 var url = "https://www.googleapis.com/upload/drive/v3/files";  
  
 request(  
 {  
 uri: url,  
 qs: {  
 uploadType: "multipart"  
 },  
 method: "POST",  
 headers: {  
 "Content-Type": "multipart/related",  
 Authorization: "Bearer " + access\_token  
 },  
 multipart: [  
 {  
 "Content-Type": "application/json; charset=UTF-8",  
 body: ***JSON***.stringify({  
 name: file.originalname  
 })  
 },  
 {  
 "Content-Type": file.mimetype,  
 body: fs.createReadStream(file.path)  
 }  
 ]  
 },  
 (error, response, body) => {  
 if (error) {  
 ***console***.info(["drive.api.js" + " : router.post.1 -> Error"]);  
 ***console***.error(error);  
 ***console***.info();  
 res.sendStatus(500);  
 }  
  
 fs.unlink(file.path);  
 res.sendStatus(200);  
 }  
 );  
});  
  
***router***.get("/user", (req, res) => {  
 if (req.session["tokens"]) {  
 res.send(200);  
 } else {  
 res.send(500);  
 }  
});  
  
module.***exports*** = ***router***;

* **Gdriveapi.ts**

import fs from 'fs';  
  
// eslint-disable-next-line @typescript-eslint/no-var-requires  
const {google} = require('googleapis');  
  
*/\*\*  
 \* Browse the link below to see the complete object returned for folder/file creation and search  
 \*  
 \** ***@link*** *https://developers.google.com/drive/api/v3/reference/files#resource  
 \*/*type PartialDriveFile = {  
 id: string;  
 name: string;  
};  
  
type SearchResultResponse = {  
 kind: 'drive#fileList';  
 nextPageToken: string;  
 incompleteSearch: boolean;  
 files: PartialDriveFile[];  
};  
  
export class GoogleDriveService {  
 private driveClient;  
  
 public constructor(clientId: string, clientSecret: string, redirectUri: string, refreshToken: string) {  
 this.driveClient = this.createDriveClient(clientId, clientSecret, redirectUri, refreshToken);  
 }  
  
 createDriveClient(clientId: string, clientSecret: string, redirectUri: string, refreshToken: string) {  
 const client = new google.auth.OAuth2(clientId, clientSecret, redirectUri);  
  
 client.setCredentials({refresh\_token: refreshToken});  
  
 return google.drive({  
 version: 'v3',  
 auth: client,  
 });  
 }  
  
 createFolder(folderName: string): Promise<PartialDriveFile> {  
 return this.driveClient.files.create({  
 resource: {  
 name: folderName,  
 mimeType: 'application/vnd.google-apps.folder',  
 },  
 fields: 'id, name',  
 });  
 }  
  
 searchFolder(folderName: string): Promise<PartialDriveFile | null> {  
 return new ***Promise***((resolve, reject) => {  
 this.driveClient.files.list(  
 {  
 q: `mimeType='application/vnd.google-apps.folder' and name='${folderName}'`,  
 fields: 'files(id, name)',  
 },  
 (err, res: { data: SearchResultResponse }) => {  
 if (err) {  
 return reject(err);  
 }  
  
 return resolve(res.data.files ? res.data.files[0] : null);  
 },  
 );  
 });  
 }  
  
 saveFile(fileName: string, filePath: string, fileMimeType: string, folderId?: string) {  
 return this.driveClient.files.create({  
 requestBody: {  
 name: fileName,  
 mimeType: fileMimeType,  
 parents: folderId ? [folderId] : [],  
 },  
 media: {  
 mimeType: fileMimeType,  
 body: fs.createReadStream(filePath),  
 },  
 });  
 }  
}

* **Index.html**

<!DOCTYPE html>  
<html lang="en">  
  
<head>  
 <base href="/">  
 <meta charset="UTF-8">  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">  
 <meta http-equiv="X-UA-Compatible" content="ie=edge">  
 <title>SS Assignment 2 - OAuth Demo</title>  
  
 <!-- Adding Bootstrap Styling Elements -->  
 <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/@fortawesome/fontawesome-free@5.15.3/css/fontawesome.min.css">  
 <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css">  
  
 <!-- Adding AngularJS Framework Elements -->  
 <script src="https://cdnjs.cloudflare.com/ajax/libs/angular.js/1.8.2/angular.min.js"></script>  
 <script src="https://cdnjs.cloudflare.com/ajax/libs/angular-route/1.8.2/angular-route.min.js"></script>  
  
</head>  
  
<body ng-app="drive">  
  
 <div class="container-fluid" ng-view></div>  
  
 <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"></script>  
 <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js"></script>  
 <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js"></script>  
 <script src="https://unpkg.com/feather-icons"></script>  
  
 <!-- Angular File Upload Directive -->  
 <script src="js/directive/ng-file-upload-shim.min.js"></script>  
 <script src="js/directive/ng-file-upload.min.js"></script>  
  
 <script src="js/app.js"></script>  
 <script src="js/app.routes.js"></script>  
 <script src="controller/app.controller.js"></script>  
 <script src="service/app.service.js"></script>  
  
 <script>  
 feather.replace();  
 </script>  
  
</body>  
  
</html>

* **Home.html**

<div class="container my-3">  
 <div class="row my-5">  
 <div class="col-md-4 mx-auto rounded">  
 <form class="col-md pt-5 pb-3 px-4 rounded border shadow text-center">  
 <div class="form-group">  
 <img class="card-img-top p-5" ngf-src="file || 'https://image.flaticon.com/icons/png/512/281/281764.png'"  
 alt="Image">  
 </div>  
 <div class="form-group" ng-show="message.message">  
 <span class="green" ng-show="message.status">{{message.message}}</span>  
 <span class="red" ng-show="!message.status">{{message.message}}</span>  
 </div>  
 <div class="form-group bg-light text-secondary rounded" for="imageUpload">  
 <label class="w-100 position-relative d-inline-block m-0 p-3" for="imageUpload">  
 <div class="div-upload">  
 <span class="glyphicon glyphicon-circle-arrow-up" aria-hidden="true"></span>  
 <span class="span-alt">Click here to select image</span>  
 </div>  
 </label>  
 <input type="file" class="form-control d-none" id="imageUpload" ngf-select ng-model="file" name="file"  
 ngf-accept="'image/\*'" required>  
 </div>  
 <div class="form-group py-3" ng-show="file">  
 <span class="text-secondary">Press the <b>Upload</b> button</span>  
 </div>  
 <div class="form-group">  
 <button type="reset" class="btn btn-block btn-secondary" ng-click="clear()">Clear</button>  
 <button type="submit" class="btn btn-block btn-primary" ng-click="upload()">Upload</button>  
 </div>  
 </form>  
 </div>  
 </div>  
</div>  
  
<ul class="nav justify-content-center mt-3">  
 <li class="nav-item">  
 <form class="nav-link">  
 <button class="btn btn-link" type="button" ng-click="logout()">Logout</button>  
 </form>  
 </li>  
</ul>

* **Login.html**

<div class="container">  
 <div class="row mt-5 justify-content-center align-self-center">  
 <div class="col-md-6 mt-5 mx-auto p-5 align-self-center shadow-lg bg-white rounded text-secondary">  
 <h2>Software Security Assignment 2</h2>  
 <div><img src="https://cdn.freelogovectors.net/wp-content/uploads/2020/11/google-drive-logo.png" alt="Google Drive Logo" class="img-fluid max-width: 20%" ></div>  
 <hr class="my-4">  
 <p class="lead text-center">  
 This web application demonstrates the <b>OAuth</b> integration for uploading images to <i>Google Drive</i>  
 </p>  
 <p>  
 <small>Use Google account to authenticate and get the access to upload files</small>  
 </p>  
  
 <a class="btn btn-lg shadow bg-white text-white p-3 pr-5 rounded font-weight-light w-100" style="background-color: #222211 **!important**;"  
 href="{{authUrl}}">  
 <i class="fab fa-google mx-3 d-inline-block"></i> Sign in with Google  
 </a>  
 </div>  
 </div>  
</div>

* **Appservice.js**

"use strict";  
  
angular.module("app.service", []).factory("AppService", [  
 "$http",  
 "$q",  
 function($http, $q) {  
 return {  
 // check user validations  
 getUser: function() {  
 var defer = $q.defer();  
  
 $http.get("/api/drive/user").then(  
 (results) => {  
 defer.resolve(results.data);  
 },  
 (err) => {  
 defer.reject(err);  
 }  
 );  
  
 return defer.promise;  
 },  
  
 // retrieve oauth url from server  
 getOAuthUrl: function() {  
 var defer = $q.defer();  
  
 $http.get("/api/drive/auth").then(  
 (results) => {  
 defer.resolve(results.data);  
 },  
 (err) => {  
 defer.reject(err);  
 }  
 );  
  
 return defer.promise;  
 },  
  
 // upload image to drive  
 uploadImage: function(image) {  
 var defer = $q.defer();  
  
 var formData = new ***FormData***();  
 formData.append("image", image);  
  
 $http  
 .post("/api/drive/upload", formData, {  
 transformRequest: angular.identity,  
 headers: {  
 "Content-Type": undefined  
 }  
 })  
 .then(  
 (result) => {  
 defer.resolve(result.data);  
 },  
 (err) => {  
 defer.reject(err);  
 }  
 );  
  
 return defer.promise;  
 },  
  
 logout: function() {  
 var defer = $q.defer();  
  
 $http.post("/api/drive/logout").then(  
 (results) => {  
 defer.resolve(results.data);  
 },  
 (err) => {  
 defer.reject(err);  
 }  
 );  
  
 return defer.promise;  
 }  
 };  
 }  
]);