

Software Security - OAuth Framework

J.P Vikum Madusanka

Faculty of Computing

Sri Lanka Institute of Information Technology

Malabe, Sri Lanka

ms20901776@my.sliit.lk +94719167131

Case: Sign-up to the PHP Webpage using Google OAuth 2.0

OAuth 2.0

OAuth is a framework simply use for authorization and give access to users to verify their identity on other 3rd party applications like web pages / mobile applications or desktop applications. With this authorization method developers are found various ways to give access to using API like Facebook/ Google/ GitHub etc. by means of OAuth, providing the user's confirmed identification.

OAuth have 3 level of defines

- Client (Resource Owner)
- Application Server (Resource)
- Authorization Server (API Server)

Client

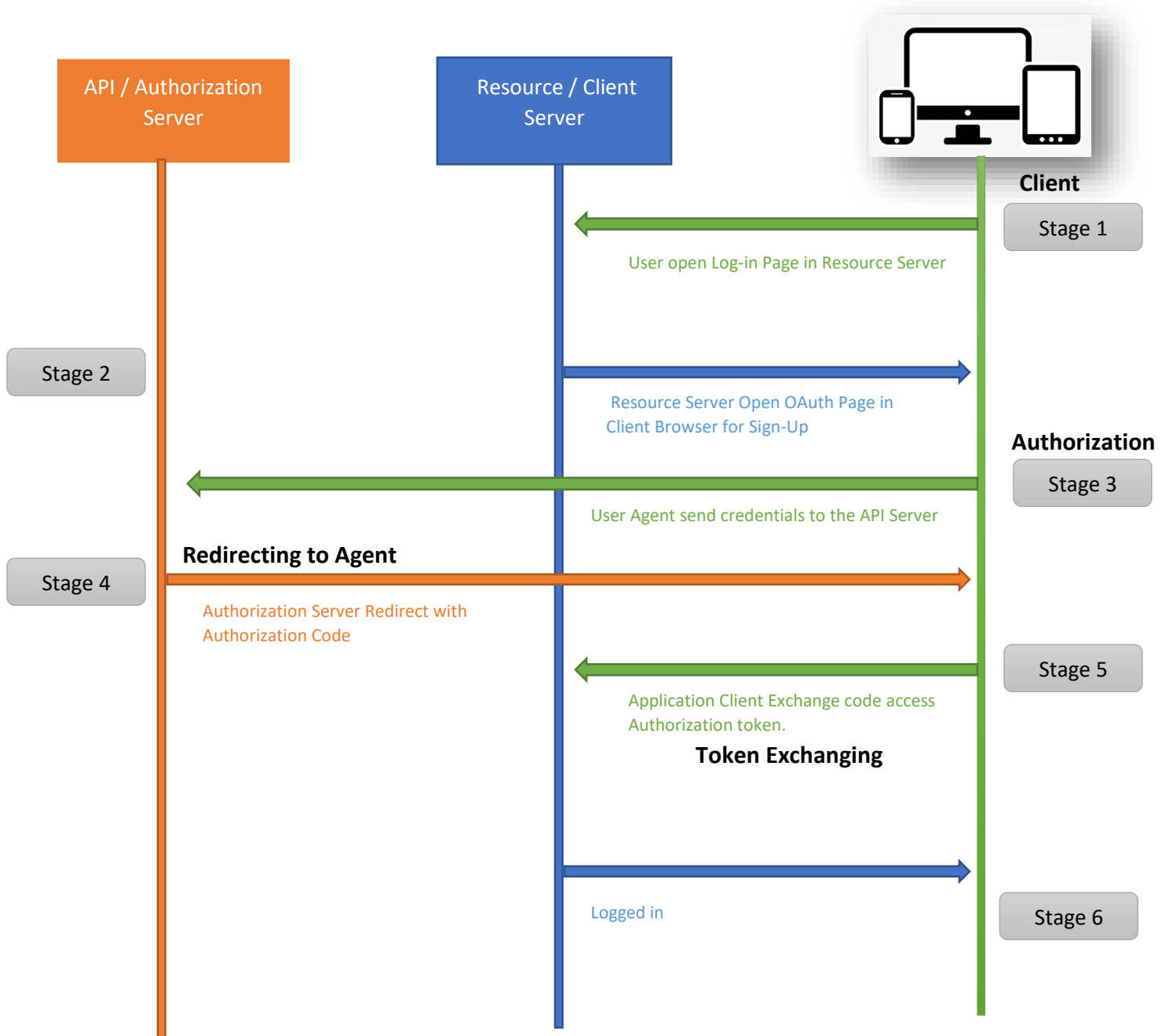
The resource owner who need to verify on desired application to access using their limited access details from trusted API party.

Resource / Client Server

The 3rd party application server that hosting application and keep User's credentials and giving access using API framework.

API server (Authorization)

Recourse server that giving limited user information access to the 3rd party applications.



- ❖ **Stage 01:** User Open Agent to Login to the Resource Server.
- ❖ **Stage 02:** Resource Server Redirect OAuth Login to User
- ❖ **Stage 03:** User send Credentials to Authorization Server.
- ❖ **Stage 04:** Authorization Server returns with code (Limited details) with redirection.
- ❖ **Stage 05:** Agent use access token to authorized Resource Server
- ❖ **Stage 06:** User Logged in

Creating Google OAuth ID

To access the Google API Service first you need to have Gmail Account or GSuite Account, then you must access Google Developer API Platform from below link.

<https://console.developers.google.com/>

The screenshot shows the 'New Project' form in the Google Cloud Platform console. At the top, there is a header with the Google APIs logo and a search bar. Below the header, the 'New Project' title is displayed. A warning message indicates that the user has 11 projects remaining in their quota and provides a link to 'Learn more' and a 'MANAGE QUOTAS' button. The 'Project name' field is required and contains the text 'SLIITMS20'. Below this, the 'Project ID' is shown as 'sliitms20' with a note that it cannot be changed later and an 'EDIT' link. The 'Location' field is also required and currently shows 'No organization' with a 'BROWSE' button. At the bottom, there are 'CREATE' and 'CANCEL' buttons.

Google APIs Search for APIs and Services

New Project

You have 11 projects remaining in your quota. Request an increase or delete projects. [Learn more](#)

[MANAGE QUOTAS](#)

Project name * ?

Project ID: sliitms20. It cannot be changed later. [EDIT](#)

Location * [BROWSE](#)

Parent organization or folder

[CREATE](#) [CANCEL](#)

Once login to the API Service we need to create a New Project & Giving a Name

After creating a New Project, we must select the privacy level of the Project. If we are using our application among our organization, we must select internal else we can select external that enabled any domain name can access it.

APIs & Services

- Dashboard
- Library
- Credentials
- OAuth consent screen**
- Domain verification
- Page usage agreements

OAuth consent screen

Choose how you want to configure and register your app, including your target users. You can only associate one app with your project.

User Type

- ☒ **Internal** ⓘ
↑ Only available to users within your organization. You will not need to submit your app for verification.
- ☐ **External** ⓘ
↑ Available to any user with a Google Account.

CREATE

Credentials

Create credentials to access your APIs

- API key
Identifies your project using a simple API key to check quota and access
- OAuth client ID**
Requests user consent so your app can access the user's data
- Service account
Enables server-to-server, app-level authentication using robot accounts
- Help me choose
Asks a few questions to help you decide which type of credential to use

Name	Creation date	Type	Client ID
sliit	May 10, 2020	Web application	511169040195-qkhp...

Service Accounts

Email	Name	Usage with all services (last 30 days)
-------	------	--

After selecting the Environment, we need to have create OAuth Client ID using Create Credential Tab

For applications that use the OAuth 2.0 protocol to call Google APIs, you can use an OAuth 2.0 client ID to generate an access token. The token contains a unique identifier. See [Setting up OAuth 2.0](#) for more information.

Application type
☒ Web application
☐ Android [Learn more](#)
☐ Chrome App [Learn more](#)
☐ iOS [Learn more](#)
☐ Other

Name ⓘ
Web client 2

Restrictions
Enter JavaScript origins, redirect URIs, or both [Learn More](#)
Origins and redirect domains must be added to the list of Authorized Domains in the [OAuth consent settings](#).

Authorized JavaScript origins
For use with requests from a browser. This is the origin URI of the client application. It can't contain a wildcard (https://*.example.com) or a path (https://example.com/subdir). If you're using a nonstandard port, you must include it in the origin URL.
https://www.example.com
Type in the domain and press Enter to add it

Authorized redirect URIs
For use with requests from a web server. This is the path in your application that users are redirected to after they have authenticated with Google. The path will be appended with the authorization code for access. Must have a protocol. Cannot contain URL fragments or relative paths. Cannot be a public IP address.
https://www.example.com
Type in the domain and press Enter to add it


[Create](#) [Cancel](#)


We need to select Application from first windows & Type the project Authorized Redirect URI for mapping specific application to Google API.

OAuth client created

The client ID and secret can always be accessed from Credentials in APIs & Services

i OAuth is limited to 100 [sensitive scope logins](#) until the [OAuth consent screen](#) is published. This may require a verification process that can take several days.

Your Client ID
511169040195-d781q2mb7ihjns1v8tk93h8dh90fip90.apps.googleusercontent.com


Your Client Secret
gP_zFjGc82KQcyh8


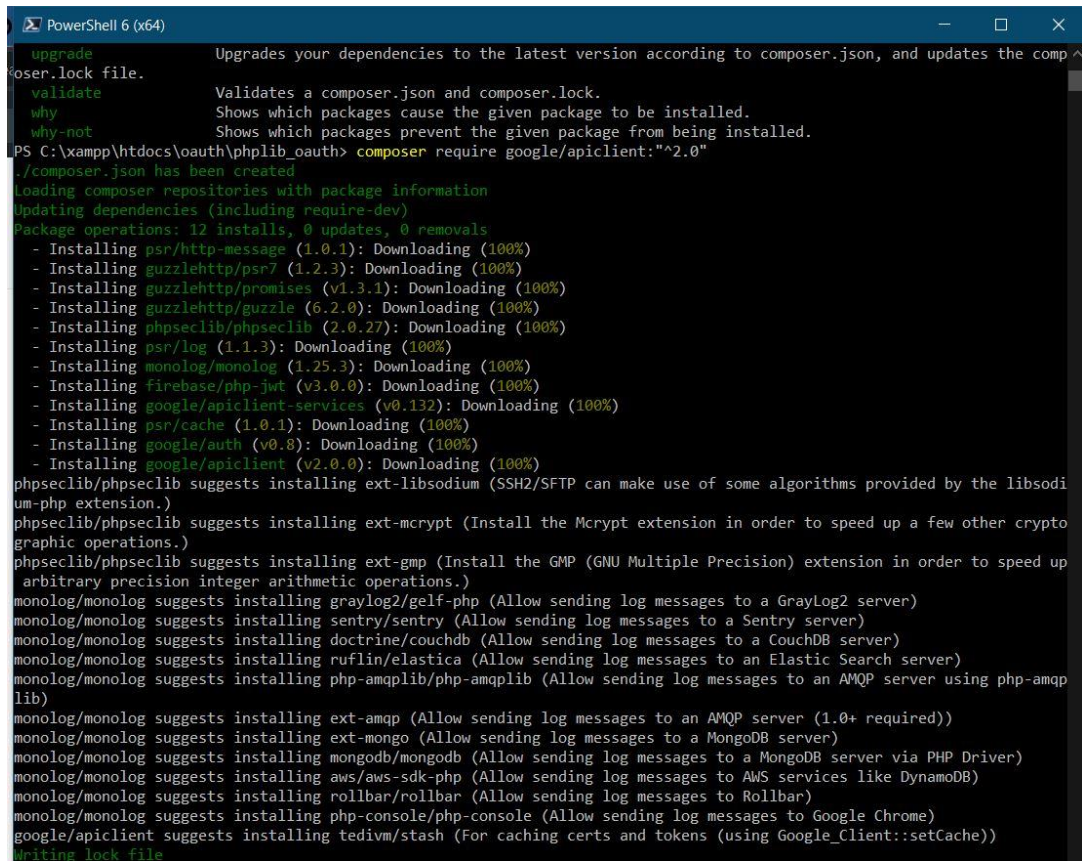
[OK](#)

Once OAuth Client Created Client ID and Secret will be show. We must save it for future use.

Then composer need to install for to the Project Folder to call the Scripts Packages.

PowerShell Command

composer require google/apiclient:"^2.0"



```
PowerShell 6 (x64)
PS C:\xampp\htdocs\oauth\phplib_oauth> composer require google/apiclient:"^2.0"

./composer.json has been created
Loading composer repositories with package information
Updating dependencies (including require-dev)
Package operations: 12 installs, 0 updates, 0 removals
- Installing psr/http-message (1.0.1): Downloading (100%)
- Installing guzzlehttp/psr7 (1.2.3): Downloading (100%)
- Installing guzzlehttp/promises (v1.3.1): Downloading (100%)
- Installing guzzlehttp/guzzle (6.2.0): Downloading (100%)
- Installing phpseclib/phpseclib (2.0.27): Downloading (100%)
- Installing psr/log (1.1.3): Downloading (100%)
- Installing monolog/monolog (1.25.3): Downloading (100%)
- Installing firebase/php-jwt (v3.0.0): Downloading (100%)
- Installing google/apiclient-services (v0.132): Downloading (100%)
- Installing psr/cache (1.0.1): Downloading (100%)
- Installing google/auth (v0.8): Downloading (100%)
- Installing google/apiclient (v2.0.0): Downloading (100%)
phpseclib/phpseclib suggests installing ext-libsodium (SSH2/SFTP can make use of some algorithms provided by the libsodium-php extension.)
monolog/monolog suggests installing ext-mcrypt (Install the Mcrypt extension in order to speed up a few other cryptographic operations.)
phpseclib/phpseclib suggests installing ext-gmp (Install the GMP (GNU Multiple Precision) extension in order to speed up arbitrary precision integer arithmetic operations.)
monolog/monolog suggests installing graylog2/gelf-php (Allow sending log messages to a GrayLog2 server)
monolog/monolog suggests installing sentry/sentry (Allow sending log messages to a Sentry server)
monolog/monolog suggests installing doctrine/couchdb (Allow sending log messages to a CouchDB server)
monolog/monolog suggests installing ruflin/elastica (Allow sending log messages to an Elastic Search server)
monolog/monolog suggests installing php-amqp/php-amqp (Allow sending log messages to an AMQP server using php-amqp lib)
monolog/monolog suggests installing ext-amqp (Allow sending log messages to an AMQP server (1.0+ required))
monolog/monolog suggests installing ext-mongo (Allow sending log messages to a MongoDB server)
monolog/monolog suggests installing mongodb/mongodb (Allow sending log messages to a MongoDB server via PHP Driver)
monolog/monolog suggests installing aws/aws-sdk-php (Allow sending log messages to AWS services like DynamoDB)
monolog/monolog suggests installing rollbar/rollbar (Allow sending log messages to Rollbar)
monolog/monolog suggests installing php-console/php-console (Allow sending log messages to Google Chrome)
google/apiclient suggests installing tedivm/stash (For caching certs and tokens (using Google_Client::setCache))
Writing lock file
```

API Client Adding to PHP Code

```
6 //OAuth Client ID
7
8
9 $google_client->setClientId('511169040195-qkhpdmr7pbddiuts0fobkursvmr3o677.apps.googleusercontent.com');
10
11 //OAuth Client Secret key
12
13 $google_client->setClientSecret('7GB8iJN49GwH3...');
14
```


Exchanging Tokens from Google API

```
7  if(isset($_GET["code"]))
8  {
9      //exchange a code for an valid token.
10
11     $token = $google_client->fetchAccessTokenWithAuthCode($_GET["code"]);
12
13     //checking errors occur during geting token
14
15     if(!isset($token['error']))
16
17     {
```

Profile Data Getting to the SESSION

```
34     //profile data and store into $_SESSION
35
36     if(!empty($data['given_name']))
37     {
38         $_SESSION['user_first_name'] = $data['given_name'];
39     }
40     if(!empty($data['family_name']))
41     {
42         $_SESSION['user_last_name'] = $data['family_name'];
43     }
44     if(!empty($data['email']))
45     {
46         $_SESSION['user_email_address'] = $data['email'];
47     }
48     if(!empty($data['gender']))
49     {
50         $_SESSION['user_image'] = $data['picture'];
51     }
52 }
```

Showing SESSION information on Page

```
<?php
if($login_button == '')
{
echo '<div class="panel-heading">Welcome User</div><div class="panel-body">';
echo '';
echo '<h3><b>Name :</b> ' .$_SESSION['user_first_name']. ' ' .$_SESSION['user_last_name']. '</h3>';
echo '<h3><b>Email :</b> ' .$_SESSION['user_email_address']. '</h3>';
echo '<h3><a href="logout.php">Logout</h3></div>';
}
else
{
    ?>
```

Logout & Redirect to Home Page

```
1  <?php
2      session_start();
3
4      session_destroy();
5
6      header('Location:index.php');
7
8      exit;
9  ?>
```


CODE / Config

```
<?php
```

```
//Google Client Library for PHP file
```

```
include('vendor/autoload.php');
```

```
$google_client = new Google_Client();
```

```
//OAuth Client ID
```

```
$google_client->setClientId('511169040195-  
qkhpdmr7pbddiuts0fobkursvmr3o677.apps.googleusercontent.com');
```

```
//OAuth Client Secret key
```

```
$google_client->setClientSecret('7GB8iJN49GwH3Xh2T5DopM6I');
```

```
//Redirect URI
```

```
$google_client->setRedirectUri('http://localhost/oauth');
```

```
$google_client->addScope('email');
```

```
$google_client->addScope('profile');
```

```
//session start
```

```
session_start();
```

```
?>
```

CODE: Index

```
<?php
```

```
//Configuration File
```

```
include('config.php');
```

```
$login_button = ";
```

```
//Google Account redirect to PHP script then this variable value has been received
```

```
if(isset($_GET["code"]))
```

```
{
```

```
//exchange a code for an valid token.
```

```
$token = $google_client->fetchAccessTokenWithAuthCode($_GET["code"]);
```

```
//checking errors occur during geting token
```

```
if(!isset($token['error']))
```

```
{
```

```
//access token requests
```

```
$google_client->setAccessToken($token['access_token']);
```

```
//Store "access_token" in $_SESSION.
```

```
$_SESSION['access_token'] = $token['access_token'];
```

```
//Create Object IN OAuth 2 class
```

```
$google_service = new Google_Service_Oauth2($google_client);
```

```
//Get user profile details
```

```
$data = $google_service->userinfo->get();
```

```
//profile data and store into $_SESSION
```

```
if(!empty($data['given_name']))
```

```
{
```

```
$_SESSION['user_first_name'] = $data['given_name'];
```

```
}
```

```
if(!empty($data['family_name']))
```

```
{
```

```
$_SESSION['user_last_name'] = $data['family_name'];
```

```
}
```

```
if(!empty($data['email']))
```

```
{
```

```
$_SESSION['user_email_address'] = $data['email'];
```

```
}
```

```
if(!empty($data['gender']))
```

```
{
```

```
$_SESSION['user_image'] = $data['picture'];
```

```
}
```

```
}
```

```
}
```

```
//check user login into system by using Google account
```

```
if(!isset($_SESSION['access_token']))
```

```
{
```

```
//Create a URL for user authorization
```

```
$login_button = '<a href="'. $google_client->createAuthUrl()."'></a>';
```

```
}
?>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<title>Sign up using Google Account</title>

<meta content='width=device-width, initial-scale=1, maximum-scale=1' name='viewport'/>


<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

<script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/4.4.1/js/bootstrap.min.js"></script>


<style>

    .frame{

        border: 1px solid black;

        padding: 20px;

        border-radius: 10px;

        padding-top: 40px;

        padding-bottom: 40px;

    }

    .btn{

        width: 100%;

    }

    .googleBtn{
```

```
        width: 75%;
        margin-left: auto;
        margin-right: auto;
        height: 40px;;
        padding: 5px;
        border-radius: 5px;
    }
    .googleBtn img{
        width: 100%;
        height: auto;
        border-radius: 5px;
    }
    .googleBtn:hover{
        opacity: 0.7;
    }
    h2{
        display: block;
        background-color: green;
        padding: 10px;
        color: white;
    }
</style>
```

```
</head>
```

```
<body>
```

```
<div class="container">
```

```
<br />
```

```
<h2 align="center">Sign up using Google Account</h2>
```

```
<br />
```

```
<div class="panel panel-default">
```


Enter Password

</div>

<button class="btn btn-primary"
type="submit">Submit</button>

<a href="<?php echo(\$google_client->createAuthUrl());
?>"><div class="googleBtn" type="submit">

</div>

</form>

</div>

</div>

</div>

<?php

}

?>

</div>

</div>

<script>

// JavaScript for disabling submissions if any invalid fields

(function() {

'use strict';

```

window.addEventListener('load', function() {
    // Fetch all the forms
    var forms = document.getElementsByClassName('needs-validation');
    // Loop
    var validation = Array.prototype.filter.call(forms, function(form) {
        form.addEventListener('submit', function(event) {
            if (form.checkValidity() === false) {
                event.preventDefault();
                event.stopPropagation();
            }
            form.classList.add('was-validated');
        }, false);
    });
}, false);
})();
</script>
</body>
</html>

```

CODE: Logout

```

<?php
session_start();

session_destroy();

header('Location:index.php');

exit;
?>

```