**Accessing GitHub User Source Code via OAuth for Threat Modeling**

This document explains the complete step-by-step process for accessing a GitHub user’s repository source code using OAuth. The goal is to fetch source code after the user logs in with GitHub and then analyze it for threat modeling.

**1. Register a GitHub OAuth App**

Go to: <https://github.com/settings/developers>

**Steps:**

* Click on **"New OAuth App"**
* Fill in:
  + **Application Name**: Your App Name (e.g., ThreatModeler)
  + **Homepage URL**: http://localhost:3000 or your app’s frontend
  + **Authorization Callback URL**: e.g., http://localhost:3000/github/callback
* After registering, you will receive:
  + Client ID
  + Client Secret

**2. Redirect User to GitHub Login**

When the user clicks "Login with GitHub", redirect them to:

https://github.com/login/oauth/authorize?client\_id=YOUR\_CLIENT\_ID&scope=repo

* scope=repo is required to read both public and private repositories.

After successful login, GitHub redirects back to your callback URL with a temporary code.

**3. Exchange Code for Access Token**

Your backend must now exchange the code for an access\_token.

**Make a POST request to:**

https://github.com/login/oauth/access\_token

**Required Parameters:**

* client\_id
* client\_secret
* code (received from GitHub)

**Required Header:**

Accept: application/json

**GitHub Response:**

{

"access\_token": "gho\_xxx...",

"scope": "repo",

"token\_type": "bearer"

}

Save the access token securely.

**4. Use the Token to Access GitHub API**

**To list user repositories:**

GET https://api.github.com/user/repos

Header: Authorization: Bearer ACCESS\_TOKEN

**To list contents of a repository:**

GET https://api.github.com/repos/USERNAME/REPO\_NAME/contents/

Header: Authorization: Bearer ACCESS\_TOKEN

**To fetch a specific file content:**

GET https://api.github.com/repos/USERNAME/REPO\_NAME/contents/PATH\_TO\_FILE

Header: Authorization: Bearer ACCESS\_TOKEN

GitHub will return:

* File name
* File type
* Encoded content (Base64)

**5. Decode and Read File Content**

If the content is Base64 encoded:

import base64

decoded\_content = base64.b64decode(file\_data['content']).decode('utf-8')

Use this decoded source code as input for your threat modeling engine.

**6. Analyze Code with Threat Modeling**

Pass the decoded source code files to your custom threat modeling logic, such as:

* STRIDE
* DREAD
* AI-powered analysis

You can analyze each file or scan the entire repo folder-by-folder.

**7. Optional Enhancements**

* Handle GitHub API pagination: GitHub returns 30 items per page by default
* Add read:user scope if you want to access user's profile data
* Add error handling for unauthorized or failed API calls

**Summary Workflow**

1. User logs in via GitHub OAuth
2. App receives code → exchanges it for access\_token
3. Use access\_token to access user's repos and files
4. Fetch, decode, and read source code files
5. Pass code to threat modeling engine

This flow is essential for your project to perform automated threat analysis on user repositories after authentication.