

گزارش کار

OAuth 2.0



امنیت شبکه

غزل کلهر (۱۹۶۷۵ - ۸۱)

فهرست مطالب

2	مقدمه
2	دستورالعمل
2	1. ایجاد برنامه OAuth در GitHub
4	2. راهاندازی سرور
5	3. طراحی صفحه ورود
7	4. احراز هویت در GitHub
8	5. احراز هویت در GitHub
9	6. دریافت اطلاعات با API سایت GitHub
10	7. ارسال درخواست‌ها به صورت خودکار
13	سوالات
15	نتیجه‌گیری

مقدمه

در این پروژه به راهاندازی یک برنامه تحت وب ساده می‌پردازیم که برای احراز هویت در آن از سایت GitHub استفاده می‌کنیم. سپس به نمایش اطلاعات پروفایل کاربر می‌پردازیم.

دستورالعمل

1. ایجاد برنامه OAuth در GitHub

در این مرحله پس از انجام مراحل گفته شده در لینک راهنمای GitHub به صفحه ثبت نام برنامه OAuth می‌رسیم. فیلد های ضروری را مطابق با آنچه که گفته شده پر می‌کنیم و نام برنامه را NS-CA2 می‌گذاریم. Homepage URL را نیز می‌گذاریم. <http://localhost:8589>

Register a new OAuth application

Application name *

NS-CA2

Something users will recognize and trust.

Homepage URL *

http://localhost:8589

The full URL to your application homepage.

Application description

Application description is optional

This is displayed to all users of your application.

Authorization callback URL *

http://localhost:8589/oauth/redirect

Your application's callback URL. Read our [OAuth documentation](#) for more information.

Register application

Cancel

پس از کلیک بر روی دکمه Register application به صفحه واقع در تصویر زیر منتقل می‌شویم. همانطور که مشاهده می‌شود Client secrets و Client ID برنامه ما مشخص شده است که در مراحل بعدی از آن‌ها استفاده خواهیم کرد.

Settings / Developer settings / NS-CA2

The screenshot shows the GitHub Developer settings interface for an application named "NS-CA2". The "General" tab is selected. On the left, there's a sidebar with "General", "Optional features", and "Advanced" options. The main area displays the following information:

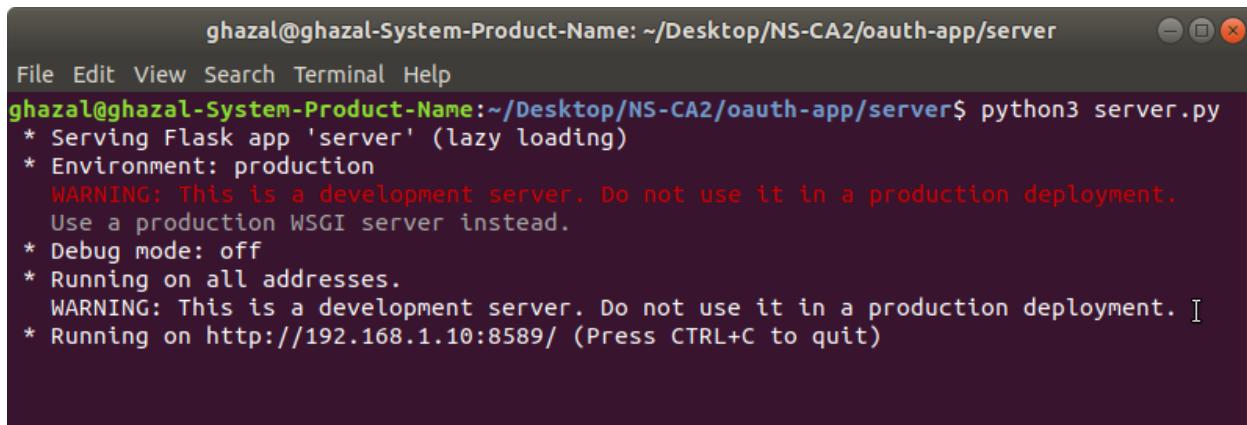
- Owner:** kalhorghazal owns this application.
- Marketplace Listing:** You can list your application in the [GitHub Marketplace](#) so that other users can discover it. A button to "List this application in the Marketplace" is present.
- Users:** 0 users. A button to "Revoke all user tokens" is available.
- Client ID:** 4d5eb5180a71cdf82b7d
- Client secrets:** A button to "Generate a new client secret". Below it, a note says: "You need a client secret to authenticate as the application to the API."

2. راه اندازی سرور

در این مرحله به اجرای دستورات گفته شده در ترمینال می‌پردازیم. در دستور اول به نصب کتابخانه Flask می‌پردازیم زیرا برای اجرای کد سرور به آن نیاز داریم (ایمپورت شده است).

```
ghazal@ghazal-System-Product-Name: ~/Desktop/server$ pip3 install flask --user
Collecting flask
  Downloading https://files.pythonhosted.org/packages/fe/b6/b4f9cb6d01e20f9cf01dcf9d3cd6c2f874b996f186f1c0b898c4a59c04/Flask-2.0.2-py3-none-any.whl (95kB)
    100% |██████████| 102kB 289kB/s
Collecting itsdangerous==2.0 (from flask)
  Downloading https://files.pythonhosted.org/packages/9c/96/26f935afba9cd6140216da5add223a0c465b99d0f112b68a4ca426441019/itsdangerous-2.0.1-py3-none-any.whl
Collecting Jinja2==3.0 (from flask)
  Downloading https://files.pythonhosted.org/packages/28/99/e5d9ec41927481e41aea8af6d16e78b5e612bca4699d417f646a9610a076/Jinja2-3.0.3-py3-none-any.whl (133kB)
    100% |██████████| 143kB 602kB/s
Collecting Werkzeug==2.0 (from flask)
  Downloading https://files.pythonhosted.org/packages/1e/73/51137805d1b8d97367a877cae4a792af14bb7ce50fb0d71af294c740cf0/Werkzeug-2.0.2-py3-none-any.whl (288kB)
    100% |██████████| 296kB 364kB/s
Collecting click==7.1.2 (from flask)
  Downloading https://files.pythonhosted.org/packages/48/58/c8aa6a8e62cc75f39fee1892c45d6b6ba684122697d7ce7d53f64f98a129/click-8.0.3-py3-none-any.whl (97kB)
    100% |██████████| 102kB 454kB/s
Collecting MarkupSafe==2.0 (from Jinja2==3.0.-flask)
  Downloading https://files.pythonhosted.org/packages/fc/d6/57ff997e564d751e340f8574836d3b636e2c14de304943836bd645fa97e/MarkupSafe-2.0.1-cp36-manylinux1_x86_64.whl
Collecting dataclasses==0.8 (from Jinja2==3.0.-flask)
  Downloading https://files.pythonhosted.org/packages/fe/75/fat5056abcf5a51bbcefef250182e50441074fd3f803f6e76451fab43/dataclasses-0.8-py3-none-any.whl
Collecting importlib-metadata; python_version < "3.8" (from click==7.1.2->flask)
  Downloading https://files.pythonhosted.org/packages/a9/01/b153a0a4cfa7a3e3f15c2cd56c7702e2cf3d9b91b359d1f1c5e59d68f4cp/importlib_metadata-4.8.3-py3-none-any.whl
Collecting typing_extensions==3.6.4; python_version < "3.8" (from importlib-metadata; python_version < "3.8" (from click==7.1.2->flask))
  Downloading https://files.pythonhosted.org/packages/05/e4/baf0031e39cf545f0c9eddb1a2ea12699b7fcba2d58e118b11753d68f0/typing_extensions-4.0.1-py3-none-any.whl
Collecting zipp==0.5 (from importlib-metadata; python_version < "3.8" (from click==7.1.2->flask))
  Downloading https://files.pythonhosted.org/packages/bd/df/d4d4974a3e3957fd1cfa3082366d7fff6e428db5f074bf6487f68e8ad/zipp-3.6.0-py3-none-any.whl
Installing collected packages: itsdangerous, MarkupSafe, Jinja2, dataclasses, Werkzeug, typing_extensions, zipp, importlib-metadata, click, flask
Successfully installed Jinja2-3.0.3 MarkupSafe-2.0.1 Werkzeug-2.0.2 click-8.0.3 dataclasses-0.8 flask-2.0.2 importlib-metadata-4.8.3 itsdangerous-2.0.1 typing_extensions-4.0.1 zipp-3.6.0
ghazal@ghazal-System-Product-Name: ~/Desktop/server$
```

دستور دوم نیز به اجرا کد سرور داده شده می‌پردازد. با توجه به تصویر پیامی مبنی بر فعالیت سرور روی پورت 8589 می‌بینیم که بیانگر اجرای موفقیت‌آمیز سرور است:



```
ghazal@ghazal-System-Product-Name:~/Desktop/NS-CA2/oauth-app/server$ python3 server.py
 * Serving Flask app 'server' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment. I
 * Running on http://192.168.1.10:8589/ (Press CTRL+C to quit)
```

3. طراحی صفحه ورود

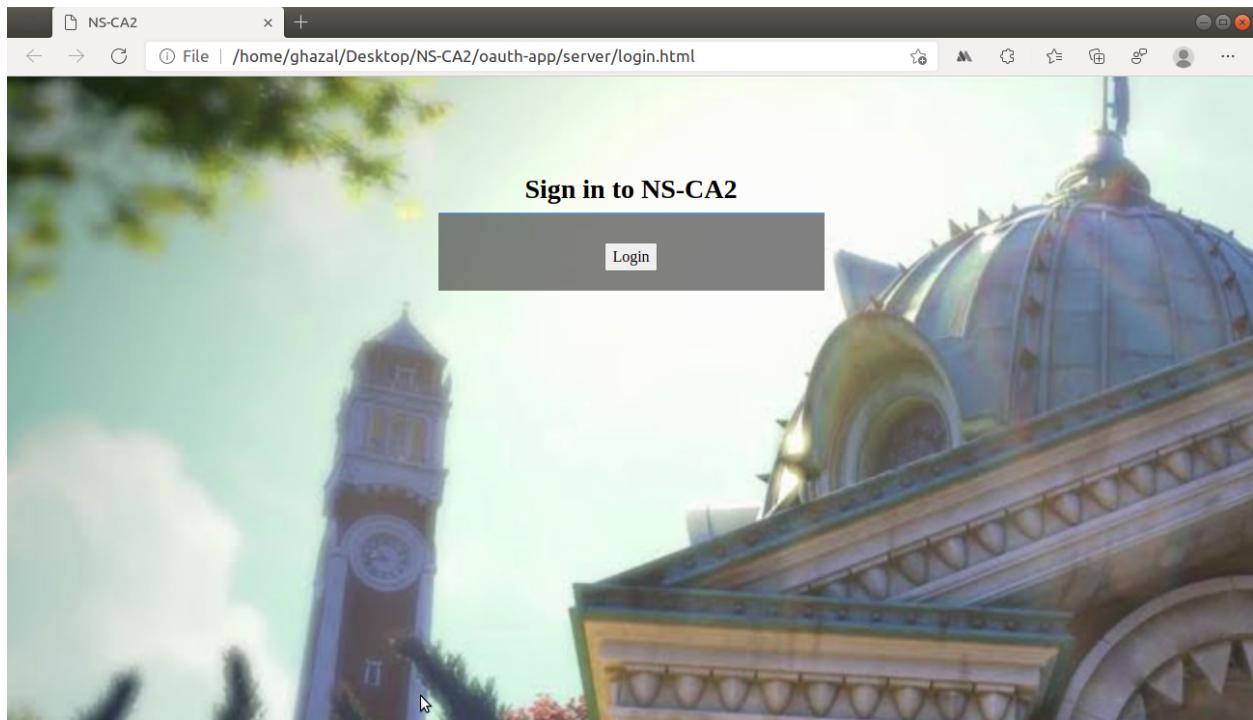
در این مرحله کد html زیر را برای طراحی صفحه ورود می‌نویسیم. با صرف نظر کردن از کدهای مربوط به style به توضیح بخش اصلی آن واقع در تگ form می‌پردازیم. آن را برابر با صفحه احراز هویت GitHub قرار می‌دهیم. متدها را برابر با GET می‌گذاریم چرا که می‌خواهیم به آن صفحه منتقل شویم. در ادامه دو تگ input داریم که در مورد اول دکمه را که از نوع submit و مقدار Login قرار می‌دهیم. در مورد دوم پارامتر client_id را با همان مقداری که در تصویر مربوط به برنامه دیدیم تنظیم می‌کنیم. همچنین نوع آن را hidden قرار می‌دهیم.

```

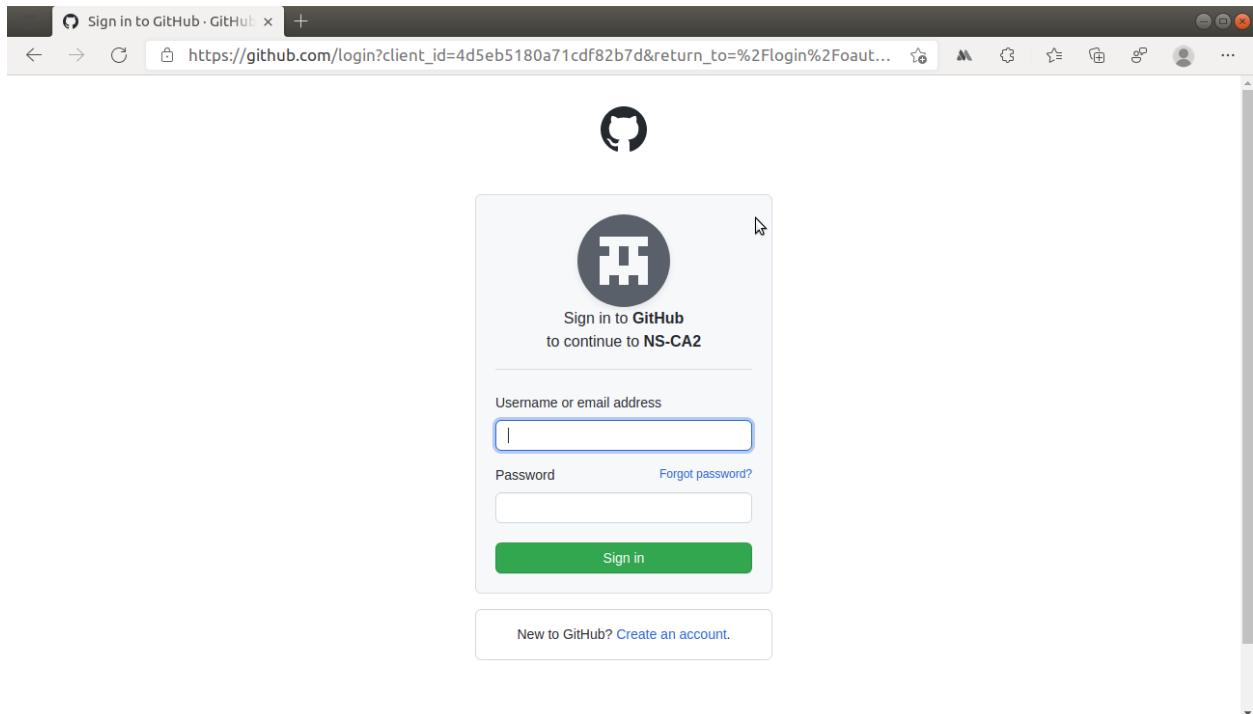
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <!-- <meta name="viewport" content="width=device-width, initial-scale=1.0"> -->
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <link rel="stylesheet" href="/login-styles.css">
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/bootstrap.min.css"
    integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPGFN9MuHof23Q9Ifjh"
    crossorigin="anonymous">
    <title>NS-CA2</title>
  </head>
  <body class="main-bg">
    <div class="login-container text-c animated flipInX">
      <h3 class="text-whiteSmoke">Sign in to NS-CA2</h3>
      <div class="container-content">
        <form class="margin-t" action="https://github.com/login/oauth/authorize" method = "GET">
          <input type = "submit" value="Login">
          <input value = "4d5eb5180a71cdf82b7d" name = "client_id" hidden>
        </form>
      </div>
    </div>
  </body>
</html>

```

در تصویر زیر خروجی کد بالا در مرورگر مشاهده می‌کیم. همانطور که دیده می‌شود یک دکمه Login روی آن وجود دارد.

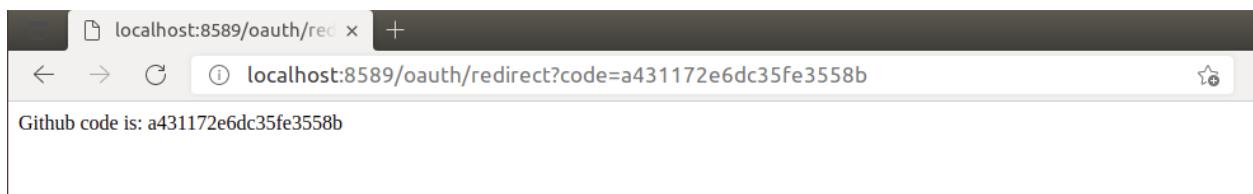


پس از کلیک بر روی دکمه Login صفحه بالا به صفحه زیر منتقل می‌شویم که در اصل صفحه احراز هویت سایت است. در URL آن نیز می‌توانیم پارامتری که حین درخواست ارسال کردیم را ببینیم.

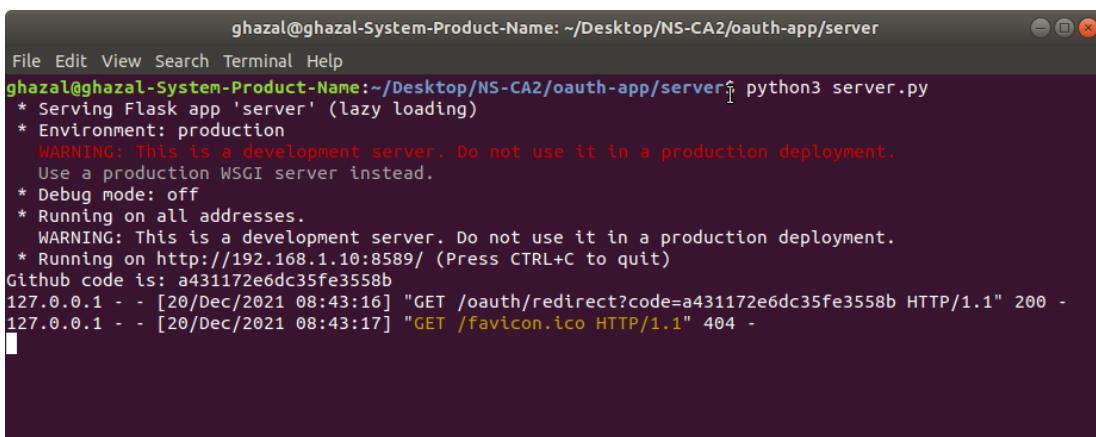


4. احراز هویت در GitHub

پس از وارد کردن اطلاعات خواسته شده در صفحه واقع در تصویر بالا و نیز کلید بر روی دکمه Sign in به صفحه زیر منتقل می‌شویم که همانی است که در callback URL حین ساخت برنامه وارد کرده بودیم. همچنین در آن یک پارامتر با مقدار code=a431172e6dc35fe3558b از طرف GitHub داده شده است.



در تصویر زیر می‌توانیم ببینیم که پارامتر code در پاسخ برگردانده شده است.



```
ghazal@ghazal-System-Product-Name: ~/Desktop/NS-CA2/oauth-app/server
File Edit View Search Terminal Help
ghazal@ghazal-System-Product-Name:~/Desktop/NS-CA2/oauth-app/server$ python3 server.py
* Serving Flask app 'server' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on all addresses.
  WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://192.168.1.10:8589/ (Press CTRL+C to quit)
Github code is: a431172e6dc35fe3558b
127.0.0.1 - - [20/Dec/2021 08:43:16] "GET /oauth/redirect?code=a431172e6dc35fe3558b HTTP/1.1" 200 -
127.0.0.1 - - [20/Dec/2021 08:43:17] "GET /favicon.ico HTTP/1.1" 404 -
```

در تصویر زیر ایمیلی که در اثر ورود ما از طرف GitHub ارسال شده است را مشاهده می‌کنیم. هدف از آن این است که کاربر از این موضوع مطلع گردد تا اگر این ورود از سوی او نبود پیگیری کند.

[GitHub] A third-party OAuth application has been added to your account ➔ [Inbox](#)

GitHub <noreply@github.com>

to me ▾

Hey kalhorghazal!

A third-party OAuth application (NS-CA2) with access to public information (read-only) was recently authorized to access your account.
Visit <https://github.com/settings/connections/applications/4d5eb5180a71cdf82b7d> for more information.

To see this and other security events for your account, visit <https://github.com/settings/security-log>

If you run into problems, please contact support by visiting <https://github.com/contact>

Thanks,
The GitHub Team

5. احراز هویت در GitHub

مطابق با تصویر زیر پارامترهای موردنیاز را در درخواست خود وارد می‌کنیم که در پاسخ آن access token gho_FpkLkmTdtMNxpZjrQ0CmS7pLfEc89X4FZ193 به عنوان gho_FpkLkmTdtMNxpZjrQ0CmS7pLfEc89X4FZ193 دریافت می‌کنیم.

POST https://github.com/login/oauth/access_token?client_id=4d5eb5180a71cdf82b7d&client_secret=6d1ecb318e300ebbea1890c7100b86ed0eec4e36&code=dfd18ab5af...

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
client_id	4d5eb5180a71cdf82b7d			
client_secret	6d1ecb318e300ebbea1890c7100b86ed0eec4e36			
code	dfd18ab5af9fd792fb1a			
Key	Value	Description		

Body Cookies Headers (17) Test Results

Status: 200 OK Time: 325 ms Size: 2.06 KB Save Response

Pretty Raw Preview Visualize Text

```
1  access_token=gho_Fpk1kmTdtMNxpZjrQ0CmS7pLfEc89X4FZ193&scope=&token_type=bearer
```

6. دریافت اطلاعات با API سایت GitHub

در این مرحله با استفاده از Token دریافت شده به دریافت اطلاعات کاربر (با شناسه kalhorghazal) به کمک API سایت GitHub می‌پردازیم. به این منظور در سرآیند درخواست Authorization را با مقدار توکن دریافت شده تنظیم می‌کنیم. در تصویر زیر می‌توانیم نتیجه حاصل از اجرای این را مشاهده کنیم:

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

Headers (7)

Key	Value	Description
User-Agent	PostmanRuntime/7.28.4	
Accept	/*	
Accept-Encoding	gzip, deflate, br	
Connection	keep-alive	
Authorization	token gho_Fpk1kmTdtMNxpZjrQ0CmS7pLfEc89X4FZ193	
Key	Value	Description

Body Cookies Headers (28) Test Results

Status: 200 OK Time: 565 ms Size: 2.65 KB Save Response

Pretty Raw Preview Visualize JSON

```

1 {
2   "login": "kalhorghazal",
3   "id": 55555804,
4   "node_id": "MDQ6VXNlcjU1NTU1ODA0",
5   "avatar_url": "https://avatars.githubusercontent.com/u/55555804?v=4",
6   "gravatar_id": "",
7   "url": "https://api.github.com/users/kalhorghazal",
8   "html_url": "https://github.com/kalhorghazal",

```

تصاویر زیر به طور کامل‌تری اطلاعات دریافت شده از پروفایل کاربر را نمایش می‌دهند.

```

1   "login": "kalhorghazal",
2   "id": 55555804,
3   "node_id": "MDQ6VXNlcjU1NTU10DA0",
4   "avatar_url": "https://avatars.githubusercontent.com/u/55555804?v=4",
5   "gravatar_id": "",
6   "url": "https://api.github.com/users/kalhorghazal",
7   "html_url": "https://github.com/kalhorghazal",
8   "followers_url": "https://api.github.com/users/kalhorghazal/followers",
9   "following_url": "https://api.github.com/users/kalhorghazal/following{/other_user}",
10  "gists_url": "https://api.github.com/users/kalhorghazal/gists{/gist_id}",
11  "starred_url": "https://api.github.com/users/kalhorghazal/starred{/owner}{/repo}",
12  "subscriptions_url": "https://api.github.com/users/kalhorghazal/subscriptions",
13  "organizations_url": "https://api.github.com/users/kalhorghazal/orgs",
14  "repos_url": "https://api.github.com/users/kalhorghazal/repos",
15  "events_url": "https://api.github.com/users/kalhorghazal/events{/privacy}",
16  "received_events_url": "https://api.github.com/users/kalhorghazal/received_events",
17  "type": "User",
18  "site_admin": false,
19
20  "name": "Ghazal Kalhor",
21  "company": "University of Tehran",
22  "blog": "https://www.linkedin.com/in/kalhorghazal/",
23  "location": null,
24  "email": null,
25  "hireable": null,
26  "bio": "Computer Engineering Student at University of Tehran",
27  "twitter_username": null,
28  "public_repos": 39,
29  "public_gists": 0,
30  "followers": 36,
31  "following": 24,
32  "created_at": "2019-09-19T18:11:18Z",
33  "updated_at": "2021-12-19T22:37:05Z"
34

```

7. ارسال درخواست‌ها به صورت خودکار

در این مرحله به تکمیل اسکریپت پایتون داده شده می‌پردازیم تا بتوانیم مراحل ۵ و ۶ را که به صورت دستی اجرا کردیم به طور خودکار انجام دهیم.

در ابتدای کد کتابخانه‌های موردنیاز برای عملیات را ایمپورت می‌کنیم. اولین کتابخانه که flask است که در کد داده شده وجود داشت. کتابخانه بعدی requests است که برای ارسال درخواست‌های مختلف POST, GET

...) به یک URL می‌توانیم از آن استفاده کنیم. کتابخانه urlparse را نیز برای تجزیه response ای که از درخواست‌ها به دست می‌آید به کار می‌بریم. از json استفاده نشد چرا که فرمت پاسخ به صورت json نبود.

```
from flask import Flask, request
import requests
from urllib.parse import urlparse
import json

try:
    from urlparse import parse_qs
except ImportError:
    from cgi import parse_qs
    urlparse.parse_qs = parse_qs

app = Flask(__name__)
```

حال به بررسی ادامه کد می‌پردازیم. در تابع oauth_redirect کد دریافت شده چاپ می‌شود. در همین متند که کد دریافت شده به ارسال درخواست‌ها می‌پردازیم. برای درخواست access token ابتدا پارامترهایی که موردنیاز است شامل client_id و client_secret و code را در قالب یک دیکشنری به صورت زوج‌های key-value در متغیری به نام tokenRequestParams ذخیره می‌کنیم. سپس متند post از کتابخانه requests را فراخوانی می‌کنیم. آرگومان اول این متند https://github.com/login/oauth/access_token است که همان URL ای است که قرار است درخواست را به آن ارسال کنیم. آرگومان دوم نیز پارامترهای درخواست است که توضیح دادیم.

خروجی حاصل از فراخوانی این متند پاسخ برگردانده شده از درخواست مذکور است که آن را در متغیری به نام tokenResponse ذخیره می‌کنیم. با توجه به اینکه ContentType پاسخ به صورت application/x-www-form-urlencoded است متند parse_qs را روی text پاسخ فراخوانی می‌کنیم تا آن را به یک دیکشنری تبدیل کند. سپس با کلید access_token به مقدار موردنظر خود دست می‌یابیم با توجه به اینکه value ها در این دیکشنری به صورت لیست هستند لازم است که اندیس صفر را نیز مشخص کنیم که مقدار را در فرمت مناسب (رشته) دریافت کنیم.

حال به ارسال درخواست به API گیت‌هاب می‌پردازیم. این درخواست از نوع get است چرا که قصد داریم داده‌ای را دریافت کنیم. در سرآیند آن نیز یک دیکشنری با یک عنصر با کلید Authorization و نیز مقدار token ارسال می‌کنیم. پاسخ این درخواست را در یک متغیر با نام profileResponse ذخیره می‌کنیم. حال کافی است که آن را در ترمینال چاپ کنیم.

```

@app.route("/oauth/redirect")
def oauth_redirect():
    code = request.args.get('code')
    print(f'Github code is: {code}')

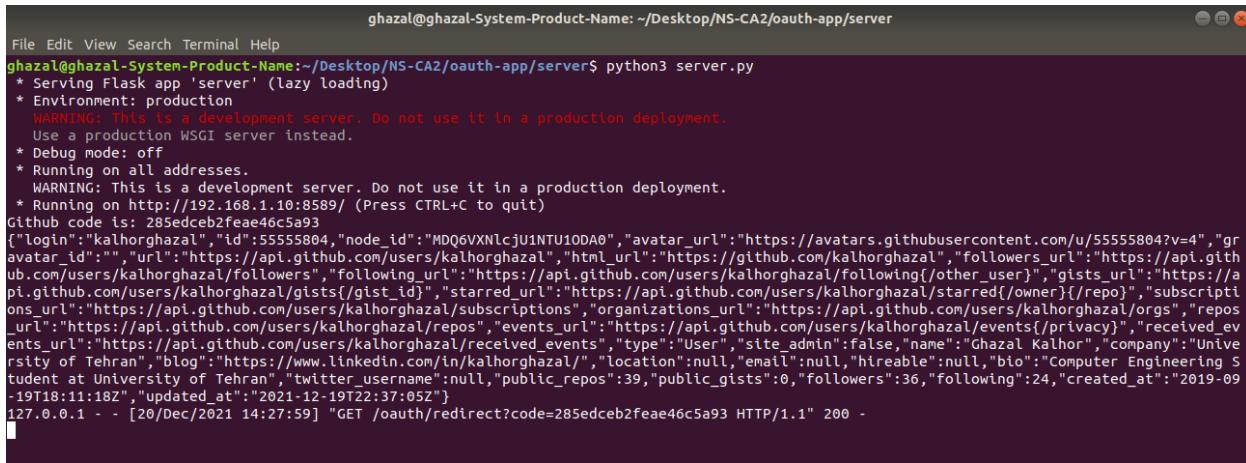
    tokenRequestParams = {'client_id': '4d5eb5180a71cdf82b7d',
                          'client_secret': '6d1ecb318e300ebbea1890c7100b86ed0eec4e36',
                          'code': str(code)}
    tokenResponse = requests.post('https://github.com/login/oauth/access_token', params=tokenRequestParams)
    token = parse_qs(tokenResponse.text)['access_token'][0]
    profileResponse = requests.get('https://api.github.com/user', headers={'Authorization': 'token ' + token})
    print(profileResponse.text)

    return f'Github code is: {code}'

if __name__ == '__main__':
    app.run(host='0.0.0.0', port = 8589)

```

در تصویر زیر خروجی حاصل اجرای کد را بعد از ورود کاربر مشاهده می‌کنیم. دقیقا مشابه با همان چیزی است که در Postman دیدیم. بنابراین تمامی مراحل کار به درستی انجام شده است.



```

File Edit View Search Terminal Help
ghazal@ghazal-System-Product-Name:~/Desktop/NS-CA2/oauth-app/server
ghazal@ghazal-System-Product-Name:~/Desktop/NS-CA2/oauth-app/server$ python3 server.py
* Serving Flask app 'server' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on all addresses.
  WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://192.168.1.10:8589/ (Press CTRL+C to quit)
Github code is: 285edceb2feae46c5a93
{"login": "kalhorghazal", "id": 55555804, "node_id": "MDQ6VXNlcjU1NTU1ODA0", "avatar_url": "https://avatars.githubusercontent.com/u/55555804?v=4", "gravatar_id": "", "url": "https://api.github.com/users/kalhorghazal", "html_url": "https://github.com/kalhorghazal", "followers_url": "https://api.github.com/users/kalhorghazal/followers", "following_url": "https://api.github.com/users/kalhorghazal/following{/other_user}", "gists_url": "https://api.github.com/users/kalhorghazal/gists{/gist_id}", "starred_url": "https://api.github.com/users/kalhorghazal/starred{/owner}{/repo}", "subscriptions_url": "https://api.github.com/users/kalhorghazal/subscriptions", "organizations_url": "https://api.github.com/users/kalhorghazal/orgs", "repos_url": "https://api.github.com/users/kalhorghazal/repos", "events_url": "https://api.github.com/users/kalhorghazal/events{/privacy}", "received_events_url": "https://api.github.com/users/kalhorghazal/received_events", "type": "User", "site_admin": false, "name": "Ghazal Kalhor", "company": "University of Tehran", "blog": "https://www.linkedin.com/in/kalhorghazal/", "location": null, "email": null, "hireable": null, "bio": "Computer Engineering Student at University of Tehran", "twitter_username": null, "public_repos": 39, "public_gists": 0, "followers": 36, "following": 24, "created_at": "2019-09-19T18:11:18Z", "updated_at": "2021-12-19T22:37:05Z"} 127.0.0.1 - - [20/Dec/2021 14:27:59] "GET /oauth/redirect?code=285edceb2feae46c5a93 HTTP/1.1" 200 -

```

سوالات

1. مزایای استفاده از روش Authorization Code Grant Type چیست؟

- از این روش به طور گسترده برای برنامه‌های موبایل و تحت وب استفاده می‌شود و در این زمینه بسیار مرسوم است.
- از آنجایی که این روش دارای مرحله اضافی مبادله Authorization Code برای access token است، یک لایه امنیتی اضافی را فراهم می‌کند که در Implicit Grant Type وجود ندارد.
- در این روش کاربر در درخواست access token دخالتی ندارد، زیرا این درخواست مستقیماً بین وب‌سایت کلاینت و سرویس احراز هویت انجام می‌شود.
- این روش توانایی احراز هویت کلاینت را دارد.
- با استفاده از refresh token به وب‌سایت مشتری دسترسی تقریباً نامحدود به داده‌های پروفایل کاربر می‌دهد.

2. در صورت استفاده از روش Authorization Code Grant Type در یک نرمافزار تلفن همراه چه ضعف‌های امنیتی متوجه این روش خواهد بود؟

- توجه داریم که نرمافزارهای تلفن همراه نمی‌توانند که یک client secret ذخیره کنند. بنابراین این شرایط یک اتکر خواهد توانست که authorization code را intercept کند.
- کاربر می‌تواند در معرض دید برنامه‌های کلاینت قرار گیرد.
- برنامه‌های کلاینت می‌توانند هر محدوده‌ای را بدون اطلاع کاربر درخواست کنند. در نتیجه سطح حمله به کاربر به طور قابل توجهی افزایش می‌یابد.
- امکان حمله phishing به کاربر وجود خواهد داشت. phishing نوعی حمله مهندسی شده اجتماعی است که اغلب برای سرقت اطلاعات کاربر شامل اعتبار ورود و شماره کارت اعتباری استفاده می‌شود.

3. آیا نوع Access Token دریافتی از انواع شناخته شده (مانند JWT) است؟ آیا میتوان این Access Token را کرد؟ در مورد نوع و روش تولید این Access Token تحقیق کنید.

مطابق با مستندات گیت‌هاب Access Token دریافتی از نوع Bearer است. Bearer Token نوع غالب Access Token هایی هستند که در OAuth 2.0 به کار می‌روند. Bearer Token در اصل یک رشته مات¹ است که برای کلاینت‌هایی که از آن استفاده می‌کنند هیچ معنای خاصی ندارد و طول آن می‌تواند متغیر باشد.

токن‌های دسترسی در احراز هویت مبتنی بر توکن به این منظور استفاده می‌شوند که به برنامه اجازه دسترسی به یک API را بدهند. هنگامی که یک توکن دسترسی دریافت کرد، هنگام درخواست API، آن توکن را به عنوان credential درج می‌کند. برای انجام این کار، باید توکن دسترسی را به عنوان یک Bearer credential مربوط به API به HTTP Authorization header ارسال کند.

لازم به ذکر است که این توکن‌ها رندوم نیستند. بر اساس کاربری که این دسترسی را به ما می‌دهد و کلاینتی که برنامه ما به آن دسترسی پیدا می‌کند ساخته می‌شود.

به عنوان مثال برای دسترسی به یک API، باید از یک Access Token استفاده کنید. توکن‌های دسترسی کوتاه مدت هستند (در تمرین ما 10 دقیقه طول عمر داشتند). ما از Bearer Token برای دریافت توکن دسترسی جدید استفاده می‌کیم. برای دریافت توکن دسترسی، این Bearer Token را به سرور احراز هویت به همراه شناسه کلاینت خود ارسال می‌کنیم. به این ترتیب سرور می‌داند که برنامه ای که از Bearer Token استفاده می‌کند، همان برنامه‌ای است که Bearer Token برای آن ایجاد شده است.

توجه داریم که این توکن‌ها توسط سرور احراز هویت تولید می‌شوند و امکان دیکود آن‌ها وجود ندارد.

4- با انتقال برنامه تولید شده توسط شما به محیط Production و استفاده از آن در محیط واقعی، حداقل یک مورد ضعف امنیتی برای برنامه شما وجود خواهد داشت. این ضعف امنیتی را شناسایی کرده و برای رفع آن راه حلی را ارائه دهید

توجه داریم که برنامه‌های موبایل و نیز برنامه‌های native هیچ راهکاری برای ذخیره client secret ندارند. روش نیز در آغاز کار برای استفاده در این گونه برنامه‌ها طراحی نشده بود. به این

¹ opaque

ترتیب چنانچه secret فاش شود، مهاجمی که یک کد معتر را به خطر می‌اندازد می‌تواند با بازیابی یک access token برای آن کاربر، دسترسی را افزایش دهد. راه حل این مشکل استفاده از PKCE است که به طور موثر نیاز به client secret را از بین می‌برد.

نتیجه‌گیری

در این پژوهه با روش Authorization Code Grant Type با استفاده از سیستم احراز هویت سایت گیت‌هاب آشنا شدیم. همچنین توانستیم یک اپلیکیشن طراحی کنیم که احراز هویت آن به کمک سایت گیت‌هاب انجام شود. همچنین با قابلیت‌های کتابخانه requests برای ارسال درخواست و دریافت پاسخ از URL‌ها آشنا شدیم.

