بسم الله الرحمن الرحیم

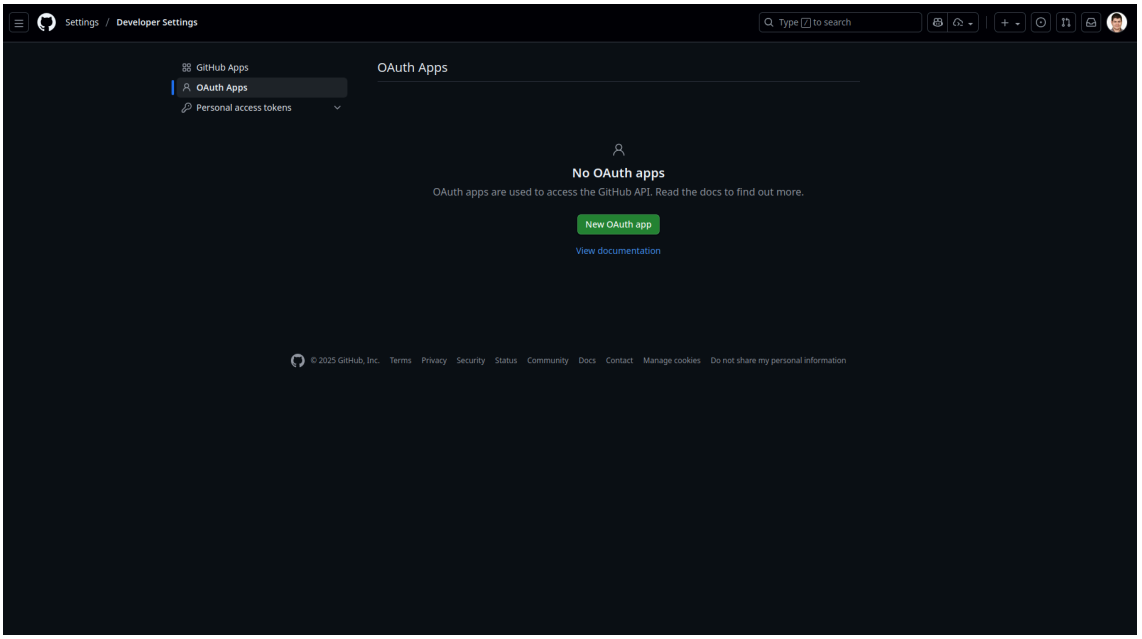
پروژه اول درس مبانی امنیت شبکه های رایانه ای

دکتر سعیدی

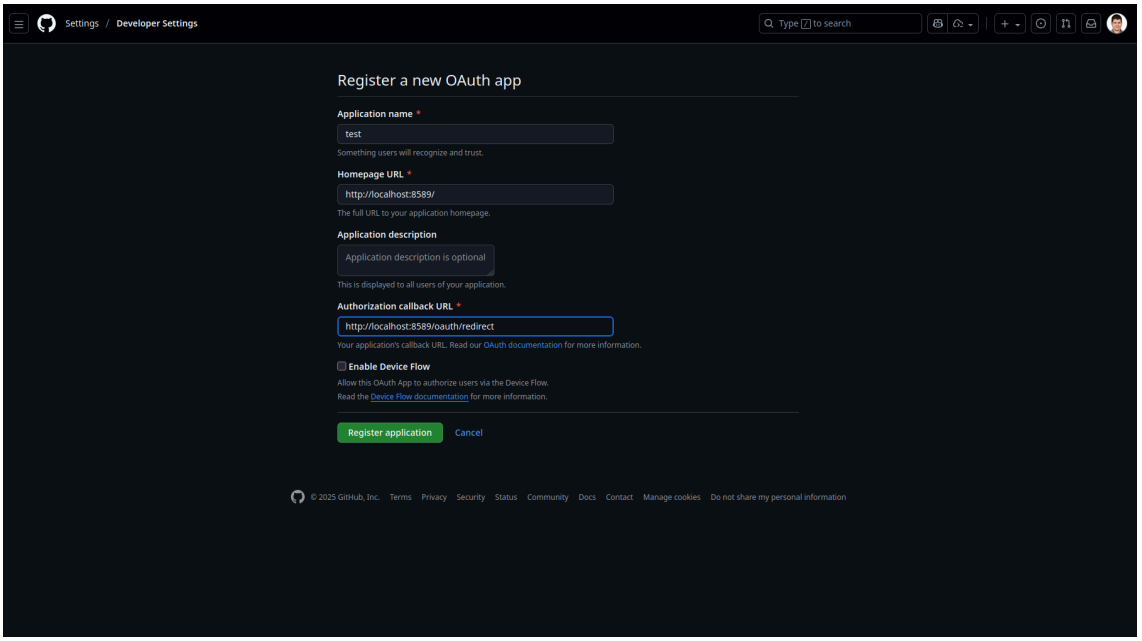
مهدی وجهی - ۸۱۰۱۰۱۵۵۸

گرفتن بهامُهر

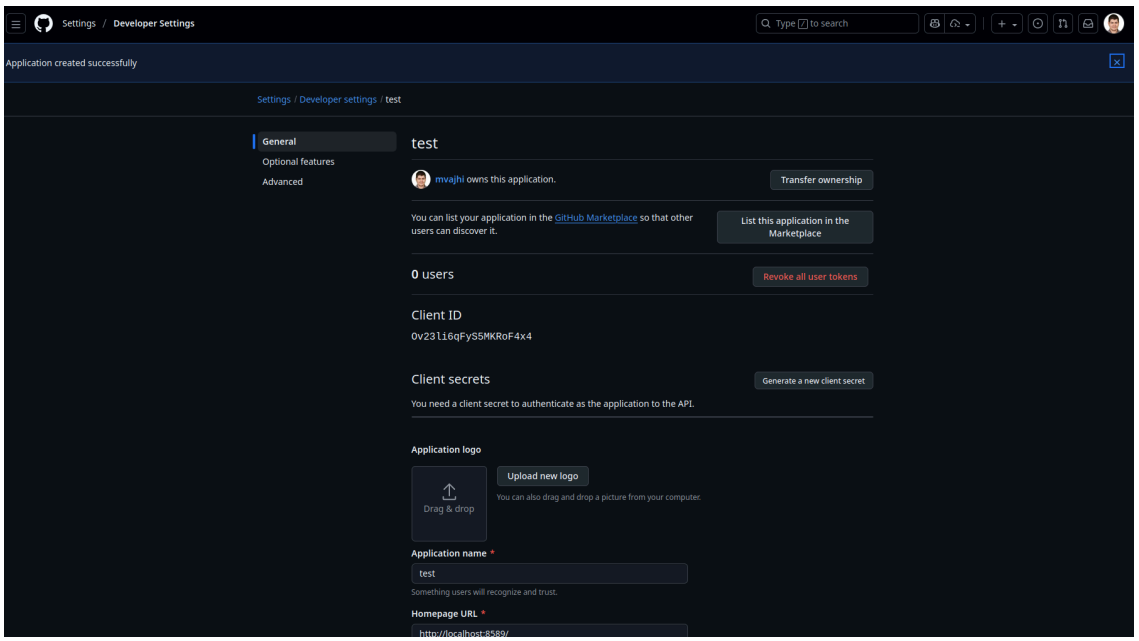
به بخش تنظیمات گیت هاب می رویم و در بخش توسعه دهندگان به بخش مربوطه می رویم.



سپس یک OAuth جدید می سازیم.



در این مرحله شناسه کارخواه و همچنین عبارت راز کارخواه را تولید و دریافت می کنیم.



زنگام صفحه ورود

زنگام صفحه ورودی را مشابه نمونه های موجود ایجاد می کنیم.

```
<!--
docs.github.com/en/apps/oauth-apps/building-oauth-apps/authorizing-oauth-apps -->
<html>
 <head>
 </head>
 <body>
   <p>
     Well, hello there!
   </p>
   <p>
     We're going to now talk to the GitHub API. Ready?
     <a
href="https://github.com/login/oauth/authorize?scope=user:email&client_id=Ov23li6qFyS5MKRoF4x4">Click here</a> to begin!
   </p>
   <p>
     If that link doesn't work, remember to provide your own <a
href="/apps/building-oauth-apps/authorizing-oauth-apps/">Client ID</a>!
   </p>
 </body>
</html>
```

و به کارساز خود معرفی می کنیم:

```python
@app.get("/login")
async def get_login_page():
    return FileResponse("./login.html")
```

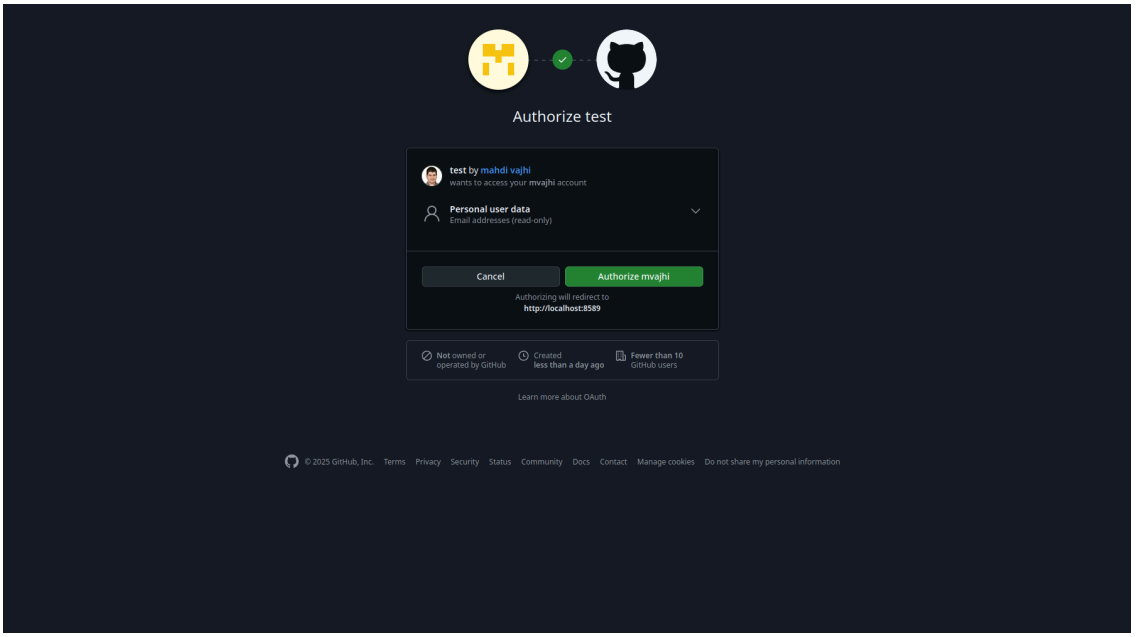
دریافت شناسه با مرورگر

وارد صفحه می شویم و سپس وارد پیوند موجود در صفحه می شویم.

Well, hello there!

We're going to now talk to the GitHub API. Ready? Click here to begin!

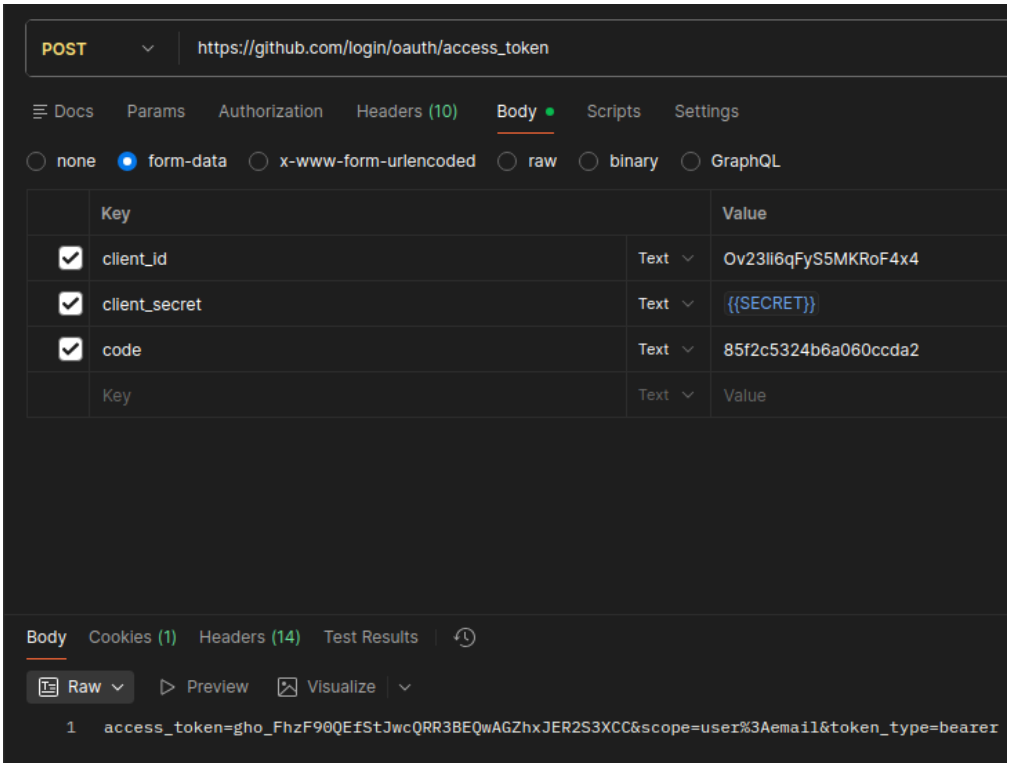
If that link doesn't work, remember to provide your own Client ID!



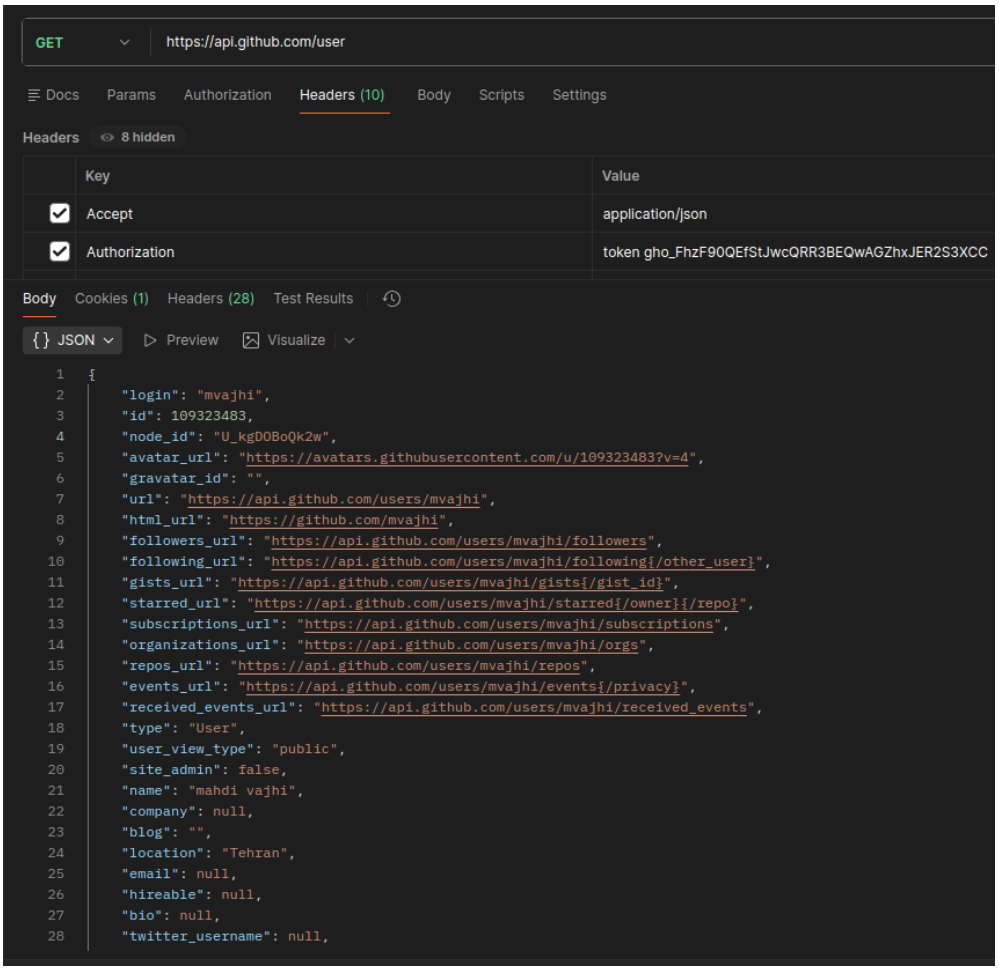
در نهایت شناسه را دریافت می کنیم.

دریافت اطلاعات کاربر توسط کارساز

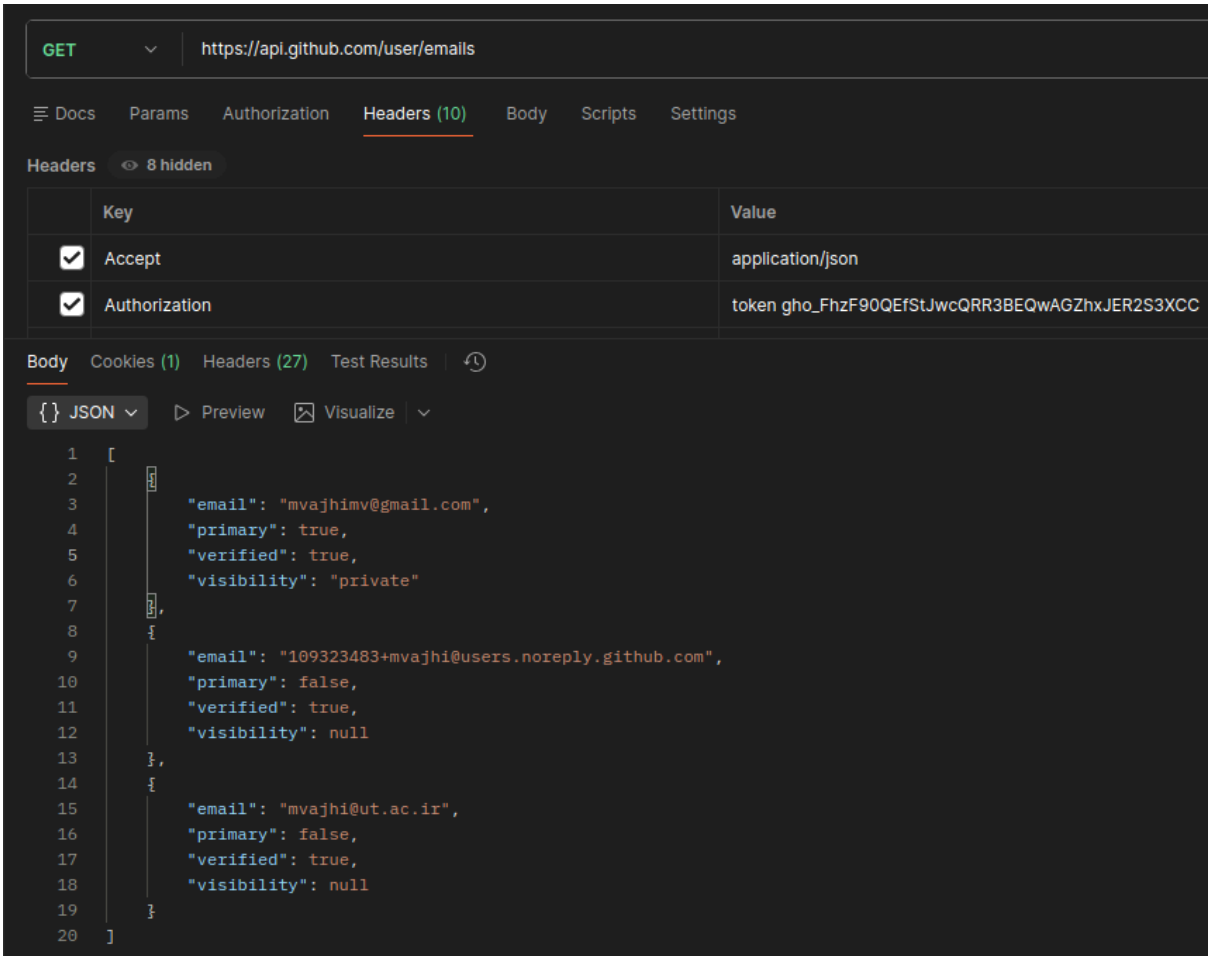
بعد از دریافت شناسه کاربر، درخواستی به گیت هاب می فرستیم تا بهامُهر دسترسی به اطلاعات کاربر را دریافت کنیم.



در این مرحله اطلاعات حساب کاربر را با درخواست زیر دریافت می کنیم.



سپس اطلاعات پست الکترونیکی کاربر را دریافت می کنیم.



اعمال تغییرات در کارساز

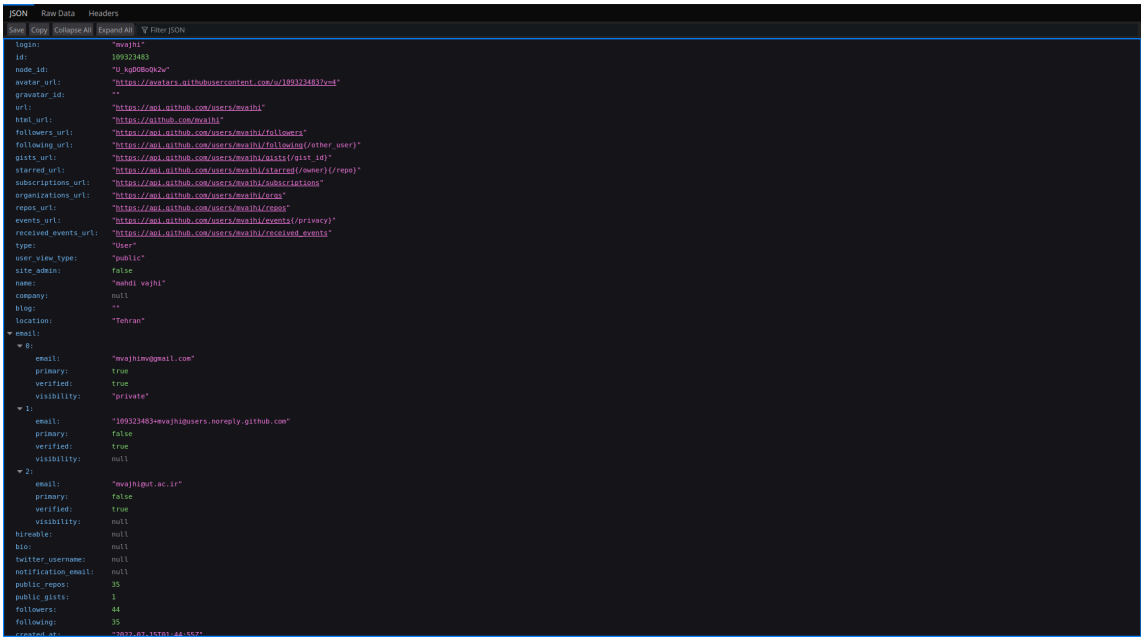
در نهایت همین درخواست ها و تبادل اطلاعات را در کارساز نیز می نویسیم. به دلیل سادگی از توضیح اضافه خودداری می کنیم.

```
@app.get("/oauth/redirect")
def oauth_redirect(code: str):
    print(f'Github code is: {code}')
    client_secret = os.getenv("SECRET")
    response = requests.post('https://github.com/login/oauth/access_token',
                json={
                    'client_id': 'Ov23li6qFyS5MKRoF4x4',
                    'client_secret': client_secret,
                    'code': code,
                }, headers={'Accept': 'application/json'})
    print(f"Response JSON:{response.json()}")
    acc_token = response.json()['access_token']
    response = requests.get('https://api.github.com/user',
           headers={'Accept': 'application/json', 'Authorization': f'token
```

```
{acc_token}'}).json()
    response_email = requests.get('https://api.github.com/user/emails',
            headers={'Accept': 'application/json', 'Authorization': f'token
{acc_token}'}).json()

    print(response)
    print(response_email)
    response['email'] = response_email
    response['Github code'] = code
    return response
```

همچنین با اجرای مجدد کارساز از صحت عملکرد آن مطمئن می شویم.



سوالات

مزایای استفاده از اعطای رمز مجوز

به دلیل این که بهامهر دسترسی مستقیم بین دو کارساز تبادل می شود امینت افزایش پیدا می کند. همچنین فرد دیگری غیر کارخواه نمی تواند اطلاعات را استعلام کند زیرا برای این کار به عبارت راز کارخواه نیاز دارد. همچنین بهامهر یکبار مصرف و با عمر کوتاه است که باعث افزایش امنیت می شود.

ضعف امنیتی اعطای اعتبار سنجی کارخواه در برنامه گوشی همراه

در این حالت لازم است که راز کارخواه در یک پیکربندی یا برنامه اجرایی ذخیره شود و مهاجم می تواند آن را استخراج کند و خود را به جای ما جا بزند.

بهامهر دسترسی

این بهامهر می تواند به دو روش ساخته شود. روش اول این است که یک رشته تصادفی تولید شود و به اطلاعات کاربر در پایگاه داده نسبت داده شود. سپس آن را به کارخواه می فرستیم و کارخواه هر دفعه آن را برای ما ارسال می کند و می توانیم با استفاده از آن اطلاعات کاربر را از پایگاه داده بخوانیم.
روش دوم استفاده از بهامهر های ساختار یافته مانند JWT است. که اطلاعات کاربر را با کلید خود امضا و رمز می کند و به کاربر ارسال می کند و کاربر در هر درخواست آن را به سمت کارساز ارسال می کند و کارساز با رمزگشایی آن می تواند داده های مورد نیاز را بازیابی کند.
در روش اول اصلا معنی برای رمزگشایی کردن وجود ندارد و برای روش دوم نیز اگر بتوانیم رمزنگاری آن را بشکنیم می توانیم داده ها را بخوانیم.
با توجه به این که OAuth ها درخواست های زیادی دریافت می کنند معقول و معمول است که از JWT استفاده کنند تا سربار حافظه را کم کنند.

آسیب پذیری کارساز

مهمترین آسیب پذیری ای که الان وجود دارد جعل درخواست از مبدا متقاطع است. به این صورت که مهاجم شناسه ورود می گیرد اما آن را به کارساز ارسال نمی کند. سپس کاربر دیگری که می خواهد وارد شود را پیدا می کند و شناسه خود را به جای شناسه ای که کاربر واقعا گرفته برای کاربر ارسال می کند و سپس کاربر با هویت مهاجم وارد سامانه می شود. راه حل آن اضافه کردن یک عبارت تصادفی و ذخیره آن در سمت کاربر و همچنین ارسال آن به کارساز احراز کننده است و در هنگام گرفتن پاسخ مقایسه کردن دو عبارت تصادفی جهت اطمینان از این که یکسان است و این پاسخ، پاسخ کارساز احراز هویت به همان کاربر است.

منابع

https://gemini.google.com/share/a46d0d798f58
https://gemini.google.com/share/a46d0d798f58
https://gemini.google.com/share/bc2693594113
https://gemini.google.com/share/fa708445de5f
https://gemini.google.com/share/d0cebd0ab84f
https://docs.github.com/en/apps/oauth-apps/building-oauth-apps/authenticating-to-the-rest-api-with-an-oauth-app