Aplicações Distribuídas
2018/19

Guião de apoio 8
Aplicações em Flask e OAuth 2

---

1. Introdução ao tema

Neste guião exploraremos o protocolo OAuth. Será implementada uma aplicação através da *framework* de desenvolvimento WEB *Flask* com autenticação e autorização OAuth.

2. Introdução ao *OAuth 2*

O protocolo OAuth utiliza HTTPS e permite a aplicações clientes (utilizadores) o acesso a recursos protegidos de outros, sem que seja necessário a obtenção das credenciais do dono do utilizador dos recursos. No entanto, é necessário que obtenha autorização prévia do dono dos recursos. É necessário haver uma terceira-parte, o servidor de autorização (autorization server), que fará a ponte entre os dois intervenientes, nomeadamente:
1) aceitando o pedido da aplicação cliente para acesso aos recursos,
2) pedindo a autorização ao dono do recurso, obtendo deste o códido de autorização (autorization code),
3) criação do token de acesso com base no código de autorização e entrega deste ao cliente.

O cliente acede aos recursos utilizando o token. Para que todo o processo de OAuth 2 funcione é necessário que a aplicação seja registada numa API de OAuth (Github, google, facebook, twitter, ...) para obtenção de um client_ID e um secret_ID e indicação do URI para onde será redireccionado o código de autorização.

A listagem abaixo é o exemplo de como conectar uma aplicação cliente em OAuth, registada na API da github.

Listagem 1 - Exemplo de utilização de OAuth

```
from requests_oauthlib import OAuth2Session
import os

# para nao suportar ligacao HTTPS
os.environ['OAUTHLIB_INSECURE_TRANSPORT'] = '1'

# Credenciais obtidas da API github no registo da aplicação
client_id = '<o id obtido da github>'
client_secret = '<o secret obtido da github>'

# Servidores da github para obtencao do authorization_code e do token
authorization_base_url = 'https://github.com/login/oauth/authorize'
token_url = 'https://github.com/login/oauth/access_token'

github = OAuth2Session(client_id)

# Pedido do authorization_code ao servidor de autorização (e dono do recuro a aceder)
authorization_url, state = github.authorization_url(authorization_base_url)
print ('Aceder ao link (via browser) para obter a autorizacao,', authorization_url)

# Obter o authorization_code do servidor vindo no URL de redireccionamento
redirect_response = input(' insira o URL devolvido no browser e cole aqui:')

# Obtencao do token
github.fetch_token(token_url, client_secret=client_secret,
                   authorization_response=redirect_response)

# Acesso a um recurso protegido
r = github.get('https://api.github.com/user')
print (r.content.decode())
```

2. Exercícios

1. Registe na GitHub (https://github.com) uma aplicação para obtenção de um client_id e secret_id. No registo da aplicação redireccione esta para o localhost e porta onde a sua aplicação está a executar.

2. Copie o programa apresentado na listagem 1, configure-o com as credenciais que obteve e execute-o. Apoiando-se na documentação sobre o módulo *requests_oauthlib* deverá entender o papel de algumas funções disponíveis nas classes disponibilizadas pelo módulo (a negrito na listagem). Para além disso deverá perceber o funcionamento do programa, identificando os passos do protocolo (indicados na introdução).

3. Crie dois programas – cliente e servidor Flask – tendo por base os slides sobre Flask. Modifique ambos, por forma a implementar autenticação mútua usando SSL, ou seja, HTTPS usando certificados digitais.

4. Altere o programa cliente para que possa comunicar com o servidor utilizando o protocolo OAuth, servindo-se do código da listagem 1 e das credenciais que obteve em 1.

5. Bibliografia e outro material de apoio

- Flask User's guide:
  http://flask.pocoo.org/docs/0.10/
- Flask API:
  http://flask.pocoo.org/docs/0.10/api/
- Módulo requests:
  http://docs.python-requests.org/en/master/
- Protocolo OAuth
  https://oauth.net
- Módulo requests-oauthlib:
  https://pypi.python.org/pypi/requests-oauthlib
  http://requests-oauthlib.readthedocs.io/en/latest/index.html