OD - Lab 4: Protocols

Exercise 1

Your tasks:

1. Create a facebook app through *Facebook for Developers* (developers.facebook.com).
    a. Select “Log In” from the menu and login using your facebook credentials
    b. Select “My Apps” from the menu and select “Create App”.
    c. Select “For Everything Else”.
    d. Give the app a name and select “Create App ID”.
2. Open the Graph API Explorer through “Tools” in the menu.
3. Select “Get Token” and “Get User Access Token”.
4. Include permission for *user_birthday* through the *API Explorer*.
5. Generate and grab your token for authorization when using cURL.
6. Construct a URL requesting your birthday from the facebook API through an appropriate endpoint
7. Use cURL to create a GET request with a custom header, making Facebook authorize your request and responding with your birth date.

The exercise will return the following message:

```
{
   “birthday”: “01/01/2000”,
   “id”: “00000000000000000”
}
```

My answers

1.-6. tasks

- Follow the steps above

7. Using cURL to create a GET request with a custom header

```
curl "https://graph.facebook.com/me?fields=id,name,birthday" -H "Authorization: Bearer <TOKEN_HERE>"
```

Exercise 1.5

In this exercise you will use wireshark to inspect network traffic.

Your tasks

1. Open wireshark from the terminal with sudo wireshark
2. Start capturing on the interface connected to the internet (likely your wifi)
3. Visit a website, for example sdu.dk
4. Stop capturing in wireshark
5. Use filtering in wireshark to see only SYN, SYN/ACK and ACK messages. Hint: You can filter based on the tcp flags (for example: `tcp.flags==0x02`)
6. Can you explain what is happening?
7. Can you find any arp messages? (filter: arp) Why/why not?

8. Can you find any dns messages? (filter: dns) Why/why not?

My answers

1.-5.

- Follow the steps above

6. Explain what you see

- This can differ, so just explain what you see
- In my case, I see a html request is being sent, and a html element being received

7. Arp messages

- Arp trafic uses mac adresses and not IP adresses

8. dns messages.

- Some nameservers are showing up in my case

Exercise 2

Your tasks:

1. Create a server in python using flask.**Hint**: Inspiration can be found in the code snippet above.More information can be found here: https://flask.palletsprojects.com/en/1.1.x/quickstart/
2. Enable the route /welcome and give it the response:“Welcome to this awesome page!”
3. Run the script and access it at http://localhost:5000/welcome.

The exercise will show the following message in the browser:

```
Welcome to this awesome page!
```

My answers

1.-2. Creating the server using flask

- Download python
- Download the flask framework

Now create the following python file with the name "main.py"

```
from flask import Flask

# This sets up the application using the Flask object from the package flask.
app = Flask(__name__)

# Using decorators the route is associated with the following method.
@app.route("/welcome/")
def hello(name):
    return "Welcome to this awesome page"

# This statement evaluates to true if this is the main python file. It starts up the Flask app on localhost with the default port 5000
if __name__ == "__main__":
    app.run(host="0.0.0.0")
```

3. Running the script

Now run the script, enter your browser and write the following

```
http://localhost:5000/welcome
```

Exercise 3

In this exercise the student is going to use the server previously created and run it in a container.

Your tasks:

1. Modify the server file from the previous exercise so that the route /welcome takes a name as a parameter, and the app responds with:“Welcome to this awesome page <*NAME*>!”where <*NAME*> is the argument.
2. Create a Dockerfile that does the following:
    a. Uses the python image as base
    b. Installs the package Flask**Hint**: This can be done in similar fashion as installing it on linux
    c. Copies the server file
    d. Runs the python server

Hint: More information on using the python image can be found on Docker Hub: https://hub.docker.com/_/python

3. Build an image from the Dockerfile. Remember to give the build a tag.
4. Run the built image and forward the container’s port to port 7000.**Hint**:The server is exposing port 5000 in the container.

Access the server running in the container at

http://localhost:7000/welcome/James

My answers

1. Modifying the server

Changing the main.py file

```
from flask import Flask

# This sets up the application using the Flask object from the package flask.
app = Flask(__name__)

# Using decorators the route is associated with the following method.
@app.route("/welcome/<name>")
def hello(name):
    return "Welcome to this awesome page " + str(name)

# This statement evaluates to true if this is the main python file. It starts up the Flask app on localhost with the default port 5000
if __name__ == "__main__":
    app.run(host="0.0.0.0")
```

2. Creating the Dockerfile

```
FROM python:alpine3.7
COPY . /app
WORKDIR /app
RUN pip install flask
EXPOSE 5000
ENTRYPOINT [ "python" ]
CMD [ "main.py" ]
```

3. Building the image

```
docker build -t exercise3 .
```

4. Running the image

```
docker run -p 5000:5000 exercise3
```

Now, enter your browser and type this:

```
http://localhost:5000/welcome/test
```

This will show the following:

```
Welcome to this awesome page test
```

Useful ressources

Dockerize your Flask Application

In this article, we'll take a look at how to dockerize a Flask application. Flask is a microframework for Python, with a basis in Werkzeug and Jinja 2. Since Docker Hub doesn't have an official Flask repository (at the time of this writing), we'll explain how to build our own.

https://runnable.com/docker/python/dockerize-your-flask-application