Let's build an API to hack - Part 2: Faking it before breaking it

Introduction

In this chapter we will be going on with the API we built in the previous exercise but this time we will be faking some things to emulate a BAC/IDOR issue for educational purposes. Later on we will create a full on login system and implement a full IDOR but for this chapter we are going to fake it by building 2 arrays and reading the list from 1 array while grabbing the details from a second array with an extra element. This might seem a bit confusing but let's dig into it to show you it does not have to be hard.

Requirements

- A patato ... Seriously though, a small VPS or spare computer with the minimal amount of RAM and disk space will do. The APIs we will be building do not require much.
- Python 3.x (https://www.python.org/downloads/)
- Flask (pip install Flask but hold off if you did not do it yet, we will be creating a python virtual env)

Let's set up

To start with, we will need to set up a virtual environment first. This is a place we can install our dependencies of a certain project on and keep them seperate from the other projects. This is very useful to keep oversight but also if you have one project that requires a certain version of an import while another project might need a much older and non-compatible version of that library.

```
mkdir "GoudAPI-BAC"
cd GoudAPI-BAC
python3 -m venv GoudAPI-BAC
```

```
mkdir GoudAPI-BAC
cd GoudAPI-BAC
py -3 -m venv GoudAPI-BAC
```

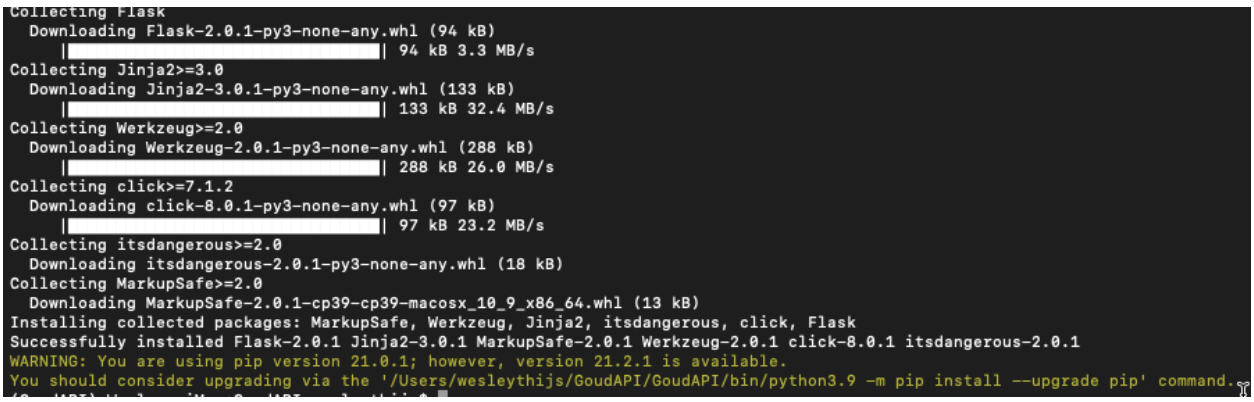
With these commands we are creating a venv (virtual enviornment) called GoudAPI-BAC which is marked by a new folder, now we have to swith to it.

```
. GoudAPI-BAC/bin/activate
```

```
GoudAPI-BAC\Scripts\activate
```

And now we can easily use pip to install flask

```
pip install Flask
```



Now that flask is installed, we can easily create our first vulnerable API.

Hello World

We will first need to write a few lines of code to tell python it should start a flask web application.

```
import flask
from flask import request, jsonify

app = flask.Flask(__name__)
app.config["DEBUG"] = True
```

These lines will tell python to import flask and to set it up with debugging enabled. app.config["DEBUG"] = True refers to a variable from the flask config. We could set up a flask config file seperatly but let's keep it all in one place for now.

We also want to import request and jsonify for later since our first API will only return static data so let's build up that static data with the following code.

```
# Create some test data for our catalog in the form of a list of dictionaries.
cheeseType = [
    {'id': 0,
     'title': 'Gouda',
     'author': 'The XSS Rat',
     'description': 'Dutch: Goudse kaas, "cheese from Gouda") is a sweet, creamy, yellow c
ows milk cheese originating from the Netherlands. ...',
     'year': '1992'},
{'id': 1,
     'title': 'Casu marzu',
     'author': 'The XSS Rat',
     'description': 'Casu martzu[1] (Sardinian pronunciation; literally rotten/putrid chee
se)is a traditional Sardinian sheep milk cheese that contains live insect larvae (maggot
s)',
     'year': '2020'},
{'id': 2,
     'title': 'Roquefort',
     'author': 'The XSS Rat',
     'description': 'Roquefort is a popular French cheese, reported to be a favourite of E
mperor Charlemagne. In France, it is called the cheese of kings and popes.',
     'year': '2020'}
]

# Create some test data with an extra record for our catalog in the form of a list of dict
ionaries.
```

```
cheeseType_detail = [
    {'id': 0,
     'title': 'Gouda',
     'author': 'The XSS Rat',
     'description': 'Dutch: Goudse kaas, "cheese from Gouda") is a sweet, creamy, yellow c
ows milk cheese originating from the Netherlands. ...',
     'year': '1992'},
{'id': 1,
     'title': 'Casu marzu',
     'author': 'The XSS Rat',
     'description': 'Casu martzu[1] (Sardinian pronunciation; literally rotten/putrid chee
se)is a traditional Sardinian sheep milk cheese that contains live insect larvae (maggot
s)',
     'year': '2020'},
{'id': 2,
     'title': 'Roquefort',
     'author': 'The XSS Rat',
     'description': 'Roquefort is a popular French cheese, reported to be a favourite of E
mperor Charlemagne. In France, it is called the cheese of kings and popes.',
     'year': '2020'},
{'id': 3,
     'title': 'Mozzarella',
     'author': 'The XSS Rat',
     'description': 'Mozzarella is a traditionally southern Italian cheese made from...',
     'year': '1998'}
]
```

So now that we have some data, we need to define a way to get that data. In the next step we will define the path /api/v1/resources/cheese/all and the possible methods. When we make a GET request on "/api/v1/resources/cheese/all" we will return the data from the previous step.

```
# A route to return all of the available entries in our catalog.
@app.route('/api/v1/resources/cheese/all', methods=['GET'])
def api_all():
    return jsonify(cheeseType)

@app.route('/api/v1/resources/cheese', methods=['GET'])
def api_id():
    # Check if an ID was provided as part of the URL.
    # If ID is provided, assign it to a variable.
    # If no ID is provided, display an error in the browser.
    if 'id' in request.args:
        id = int(request.args['id'])
    else:
        return "Error: No id field provided. Please specify an id."

    # Create an empty list for our results
    results = []
```

```
    # Loop through the data and match results that fit the requested ID.
    # IDs are unique, but other fields might return many results
    for cheese in cheeseType_detail:
        if cheeseType_detail['id'] == id:
            results.append(cheese)

    # Use the jsonify function from Flask to convert our list of
    # Python dictionaries to the JSON format.
    return jsonify(results)
```

all that is left is for us to start up our application but as of now flask does not know how to start yet so let's tell it. We will also be using another port in this case because port 5000 (the default port) will already be taken from our previous script.

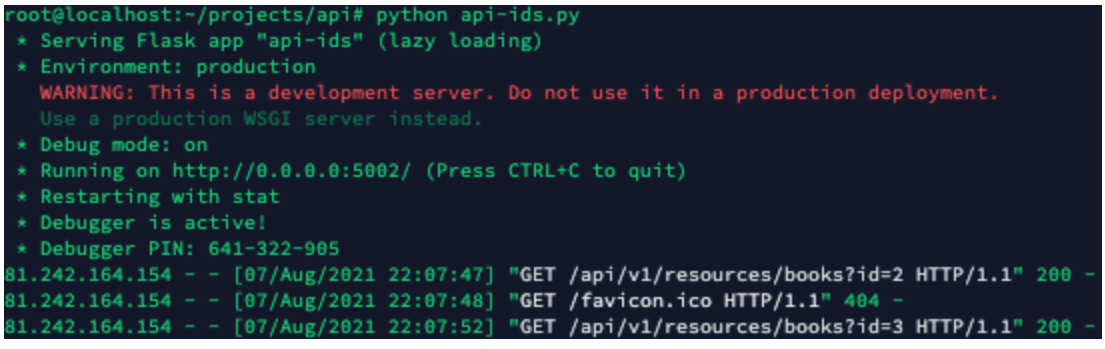
```
app.run(host="0.0.0.0",port="5001")
```

Combine all of this into one script and it will start up when we ask it, one last thing i want to highlight here is the host="0.0.0.0" argument. We do not need this but if we do not give it this argument, the API will only be accesible from localhost. The default port for flask will be 5000 but we told our script to run on port 5001 so make sure nothing else is running on that port.

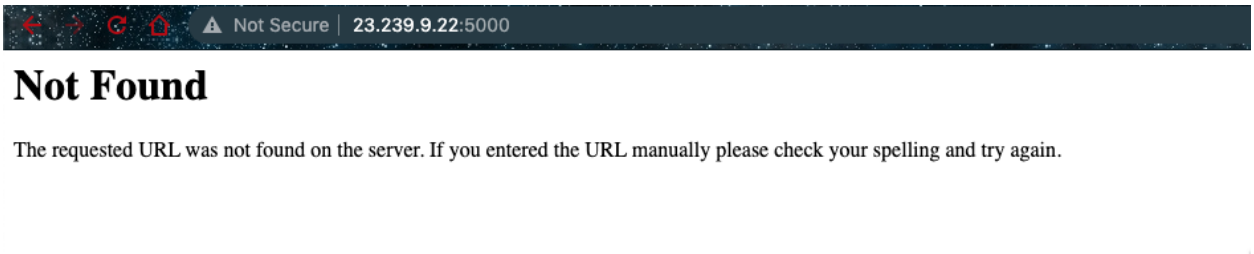
Combine all of that into a file and we are ready to go.

```
python training1.py
```

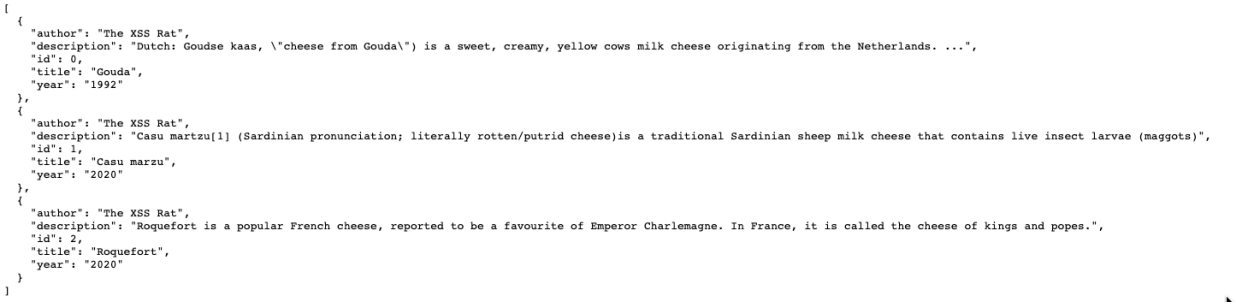
replace training1.py with whatever you named your file.



And with that, we should be able to surf to our VPS' IP adres on port 5001 and see what our flask app is saying.



To get some data, we need to surf to the path we defined earlier. "/api/v1/resources/cheese/all"



Make sure this works as this will be our basefile for the rest of the exercises.

We can also surf to "/api/v1/resources/cheese?id=3" to find our hidden type of mozzarella cheese. This would constitute a Broken Access Control issues since we are not logged in but we are able to view data we should not be able to see according to our list call to "/api/v1/resources/cheese/all".

Extra: Full code

```
import flask
from flask import request, jsonify

app = flask.Flask(__name__)
app.config["DEBUG"] = True

# Create some test data for our catalog in the form of a list of dictionaries.
cheeseType = [
    {'id': 0,
     'title': 'Gouda',
```

```
     'author': 'The XSS Rat',
     'description': 'Dutch: Goudse kaas, "cheese from Gouda") is a sweet, creamy, yellow c
ows milk cheese originating from the Netherlands. ...',
     'year': '1992'},
{'id': 1,
     'title': 'Casu marzu',
     'author': 'The XSS Rat',
     'description': 'Casu martzu[1] (Sardinian pronunciation; literally rotten/putrid chee
se)is a traditional Sardinian sheep milk cheese that contains live insect larvae (maggot
s)',
     'year': '2020'},
{'id': 2,
     'title': 'Roquefort',
     'author': 'The XSS Rat',
     'description': 'Roquefort is a popular French cheese, reported to be a favourite of E
mperor Charlemagne. In France, it is called the cheese of kings and popes.',
     'year': '2020'}
]

# Create some test data with an extra record for our catalog in the form of a list of dict
ionaries.
cheeseType_detail = [
    {'id': 0,
     'title': 'Gouda',
     'author': 'The XSS Rat',
     'description': 'Dutch: Goudse kaas, "cheese from Gouda") is a sweet, creamy, yellow c
ows milk cheese originating from the Netherlands. ...',
     'year': '1992'},
{'id': 1,
     'title': 'Casu marzu',
     'author': 'The XSS Rat',
     'description': 'Casu martzu[1] (Sardinian pronunciation; literally rotten/putrid chee
se)is a traditional Sardinian sheep milk cheese that contains live insect larvae (maggot
s)',
     'year': '2020'},
{'id': 2,
     'title': 'Roquefort',
     'author': 'The XSS Rat',
     'description': 'Roquefort is a popular French cheese, reported to be a favourite of E
mperor Charlemagne. In France, it is called the cheese of kings and popes.',
     'year': '2020'},
{'id': 3,
     'title': 'Mozzarella',
     'author': 'The XSS Rat',
     'description': 'Mozzarella is a traditionally southern Italian cheese made from...',
     'year': '1998'}
]

# A route to return all of the available entries in our catalog.
@app.route('/api/v2/resources/cheese/all', methods=['GET'])
def api_all():
    return jsonify(cheeseType)

@app.route('/api/v1/resources/cheese', methods=['GET'])
```

```
def api_id():
    # Check if an ID was provided as part of the URL.
    # If ID is provided, assign it to a variable.
    # If no ID is provided, display an error in the browser.
    if 'id' in request.args:
        id = int(request.args['id'])
    else:
        return "Error: No id field provided. Please specify an id."

    # Create an empty list for our results
    results = []

    # Loop through the data and match results that fit the requested ID.
    # IDs are unique, but other fields might return many results
    for cheese in cheeseType_detail:
        if cheeseType_detail['id'] == id:
            results.append(cheese)

    # Use the jsonify function from Flask to convert our list of
    # Python dictionaries to the JSON format.
    return jsonify(results)

app.run(host="0.0.0.0",port="5001")
```