Integrating Security and Automation

The following topics will briefly be discussed in this lab:

- Automation frameworks and techniques
- Automating existing security testing
- Security testing with an existing automation framework

HTTP mock server

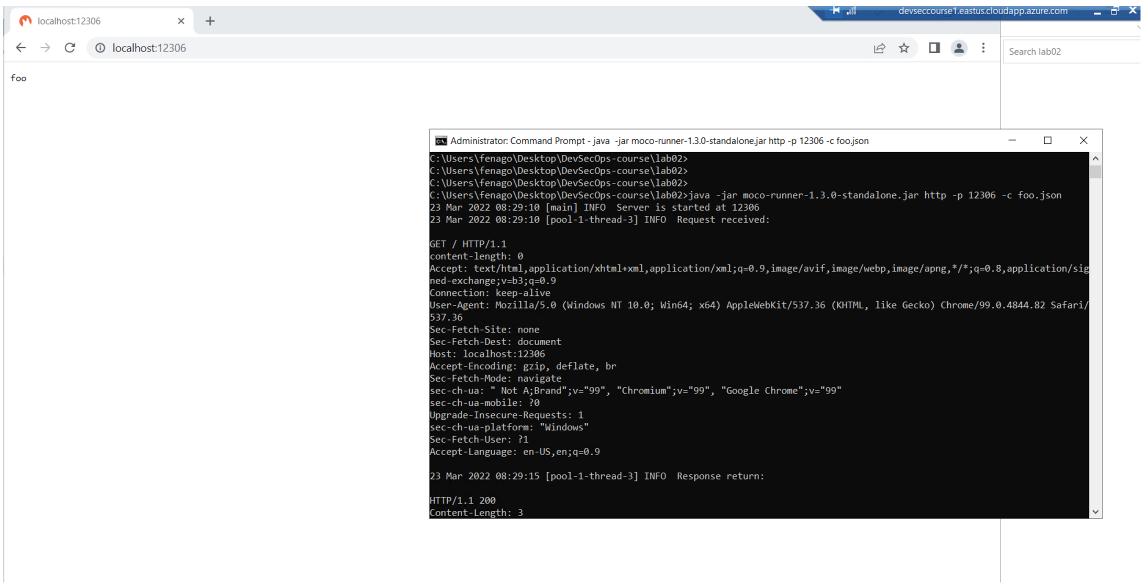
In this task, we will setup and run "moco" https://github.com/dreamhead/moco .

`moco-runner-1.3.0-standalone.jar` has been downloaded already in `lab02` folder.

Run following commands in the cmd to start mock server:

```
cd C:\Users\fenago\Desktop\DevSecOps-course\lab02

java -jar moco-runner-1.3.0-standalone.jar http -p 12306 -c foo.json
```

Open http://localhost:12306 in Chrome after starting mock server.

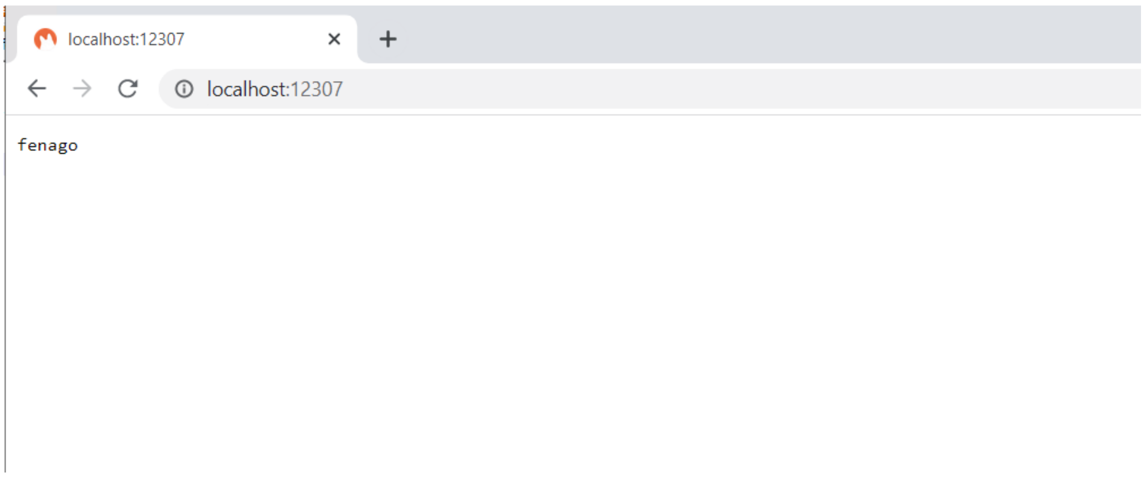

localhost:12306
localhost:12306
devseccourse1.eastus.cloudapp.azure.com
Search lab02
Administrator: Command Prompt - java -jar moco-runner-1.3.0-standalone.jar http -p 12306 -c foo.json

```
foo
```

```
C:\Users\fenago\Desktop\DevSecOps-course\lab02>
C:\Users\fenago\Desktop\DevSecOps-course\lab02>
C:\Users\fenago\Desktop\DevSecOps-course\lab02>
C:\Users\fenago\Desktop\DevSecOps-course\lab02>java -jar moco-runner-1.3.0-standalone.jar http -p 12306 -c foo.json
23 Mar 2022 08:29:10 [main] INFO  Server is started at 12306
23 Mar 2022 08:29:10 [pool-1-thread-3] INFO  Request received:

GET / HTTP/1.1
content-length: 0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/sig
ned-exchange;v=b3;q=0.9
Connection: keep-alive
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/99.0.4844.82 Safari/
537.36
Sec-Fetch-Site: none
Sec-Fetch-Dest: document
Host: localhost:12306
Accept-Encoding: gzip, deflate, br
Sec-Fetch-Mode: navigate
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="99", "Google Chrome";v="99"
sec-ch-ua-mobile: ?0
Upgrade-Insecure-Requests: 1
sec-ch-ua-platform: "Windows"
Sec-Fetch-User: ?1
Accept-Language: en-US,en;q=0.9

23 Mar 2022 08:29:15 [pool-1-thread-3] INFO  Response return:

HTTP/1.1 200
Content-Length: 3
```

Exercises

1. Update `foo.json` to return your name in response as show below:
2. Change port to 12307 and restart the server.



fenago

Behavior-driven development testing frameworks

`Behave` has been installed already in the lab environment. https://github.com/behave/behave

Create a directory called "features/". In that directory create a file called "example.feature" containing:

```
# -- FILE: features/example.feature
Feature: Showing off behave

  Scenario: Run a simple test
    Given we have behave installed
     When we implement 5 tests
     Then behave will test them for us!
```

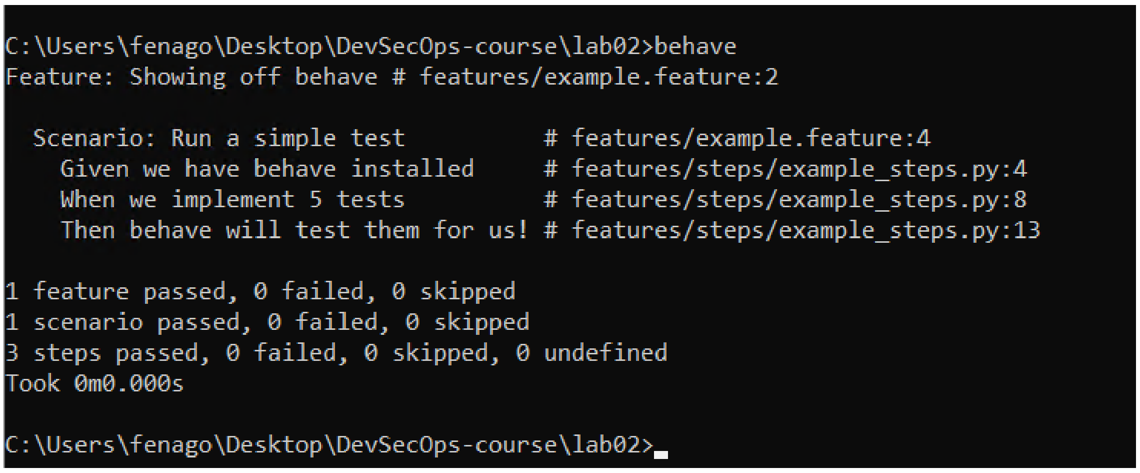
Make a new directory called "features/steps/". In that directory create a file called "example_steps.py" containing:

```
# -- FILE: features/steps/example_steps.py
from behave import given, when, then, step

@given('we have behave installed')
def step_impl(context):
    pass

@when('we implement {number:d} tests')
def step_impl(context, number):  # -- NOTE: number is converted into integer
    assert number > 1 or number == 0
    context.tests_count = number

@then('behave will test them for us!')
def step_impl(context):
    assert context.failed is False
    assert context.tests_count >= 0
```

Run behave by running following command in the terminal: `behave`

Output:

```
Feature: Showing off behave # features/example.feature:2

  Scenario: Run a simple test          # features/example.feature:4
    Given we have behave installed     # features/steps/example_steps.py:4
    When we implement 5 tests          # features/steps/example_steps.py:8
    Then behave will test them for us! # features/steps/example_steps.py:13

1 feature passed, 0 failed, 0 skipped
1 scenario passed, 0 failed, 0 skipped
3 steps passed, 0 failed, 0 skipped, 0 undefined
```



```
C:\Users\fenago\Desktop\DevSecOps-course\lab02>behave
Feature: Showing off behave # features/example.feature:2

  Scenario: Run a simple test          # features/example.feature:4
    Given we have behave installed     # features/steps/example_steps.py:4
    When we implement 5 tests          # features/steps/example_steps.py:8
    Then behave will test them for us! # features/steps/example_steps.py:13

1 feature passed, 0 failed, 0 skipped
1 scenario passed, 0 failed, 0 skipped
3 steps passed, 0 failed, 0 skipped, 0 undefined
Took 0m0.000s

C:\Users\fenago\Desktop\DevSecOps-course\lab02>
```

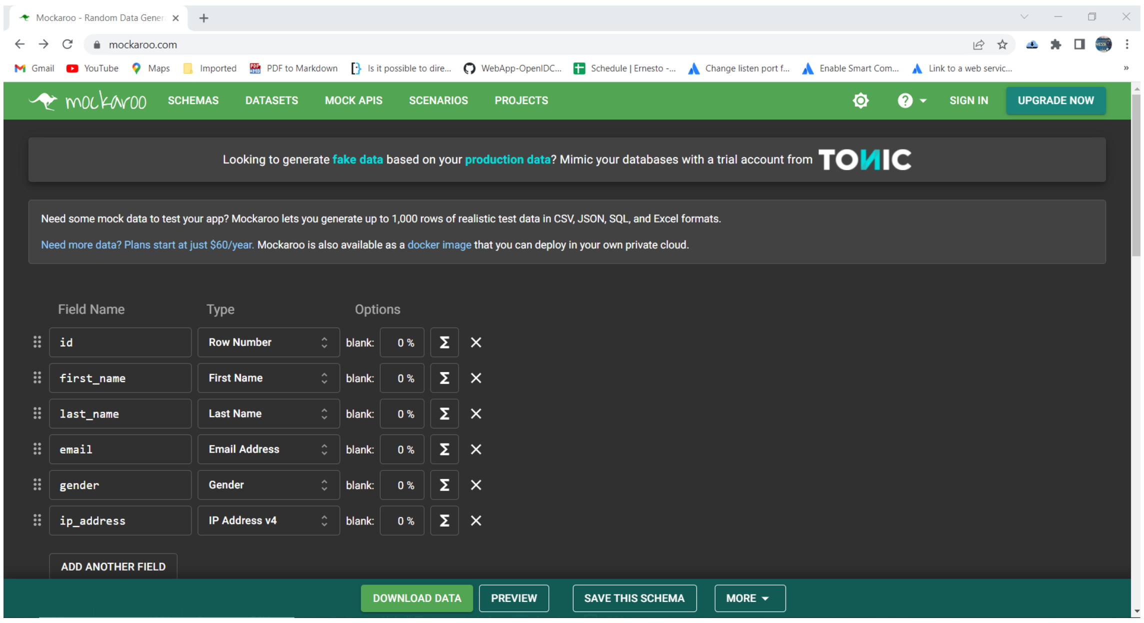
In the coming labs, we will demostrate the uses of Robot Framework to achieve the security automation testing.

Testing data generators

Here are some tools that can generate testing data based on the user-defined data type or format such as (date, address, ID, numeric data or strings):

Mockaroo

Open https://mockaroo.com/ in browser and click **Preview Data** to get generated data:



Looking to generate fake data based on your production data? Mimic your databases with a trial account from TONIC

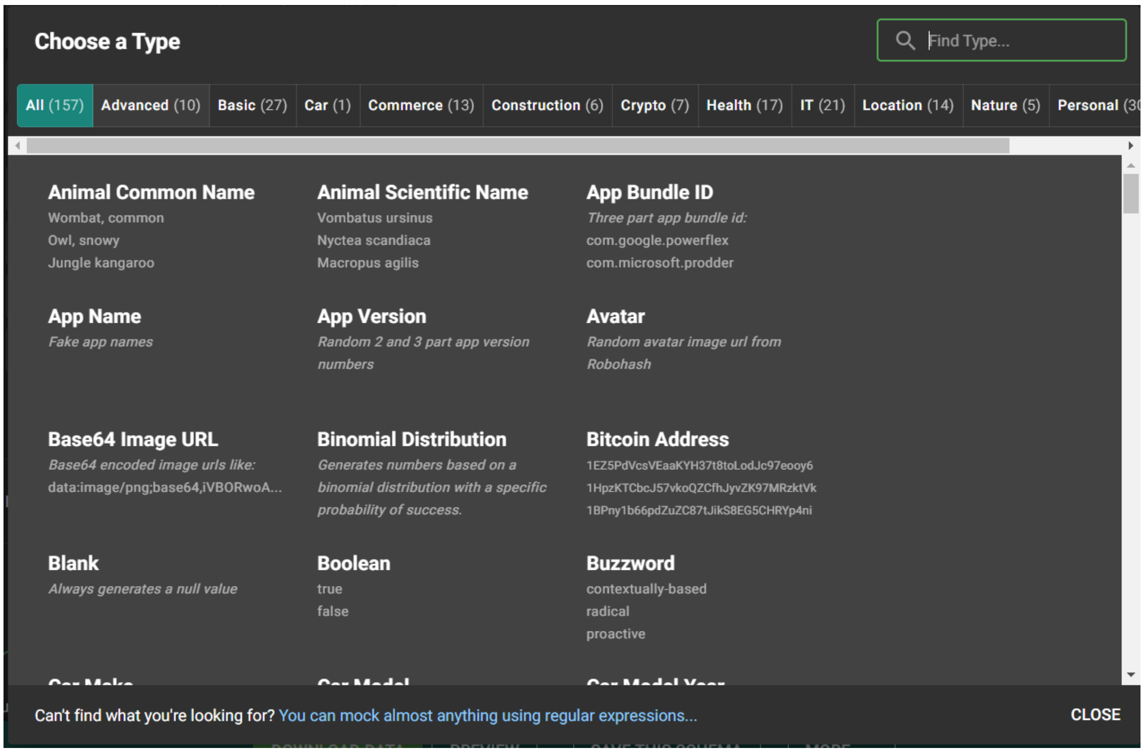
Need some mock data to test your app? Mockaroo lets you generate up to 1,000 rows of realistic test data in CSV, JSON, SQL, and Excel formats.

Need more data? Plans start at just $60/year. Mockaroo is also available as a docker image that you can deploy in your own private cloud.

Field Name	Type	Options
id	Row Number	blank: 0 %
first_name	First Name	blank: 0 %
last_name	Last Name	blank: 0 %
email	Email Address	blank: 0 %
gender	Gender	blank: 0 %
ip_address	IP Address v4	blank: 0 %

Task:

1. Remove `ip_address` field by clicking `X`.
2. Add three new fields by clicking `Add Another Field` button.
3. Click **Type** dropdown and select different Type from menu for all new fields:



Animal Common Name
Wombat, common
Owl, snowy
Jungle kangaroo

Animal Scientific Name
Vombatus ursinus
Nyctea scandiaca
Macropus agilis

App Bundle ID
Three part app bundle id:
com.google.powerflex
com.microsoft.prodder

App Name
Fake app names

App Version
Random 2 and 3 part app version numbers

Avatar
Random avatar image url from Robohash

Base64 Image URL
Base64 encoded image urls like:
data:image/png;base64,iVBORwoA...

Binomial Distribution
Generates numbers based on a binomial distribution with a specific probability of success.

Bitcoin Address
1EZ5PdVcsVEaaKYH37t8toLodJc97eooy6
1HpzKTCbcJ57vkoQZCfhJyvZK97MRzktVk
1BPny1b66pdZuZC87tJikS8EG5CHRYp4ni

Blank
Always generates a null value

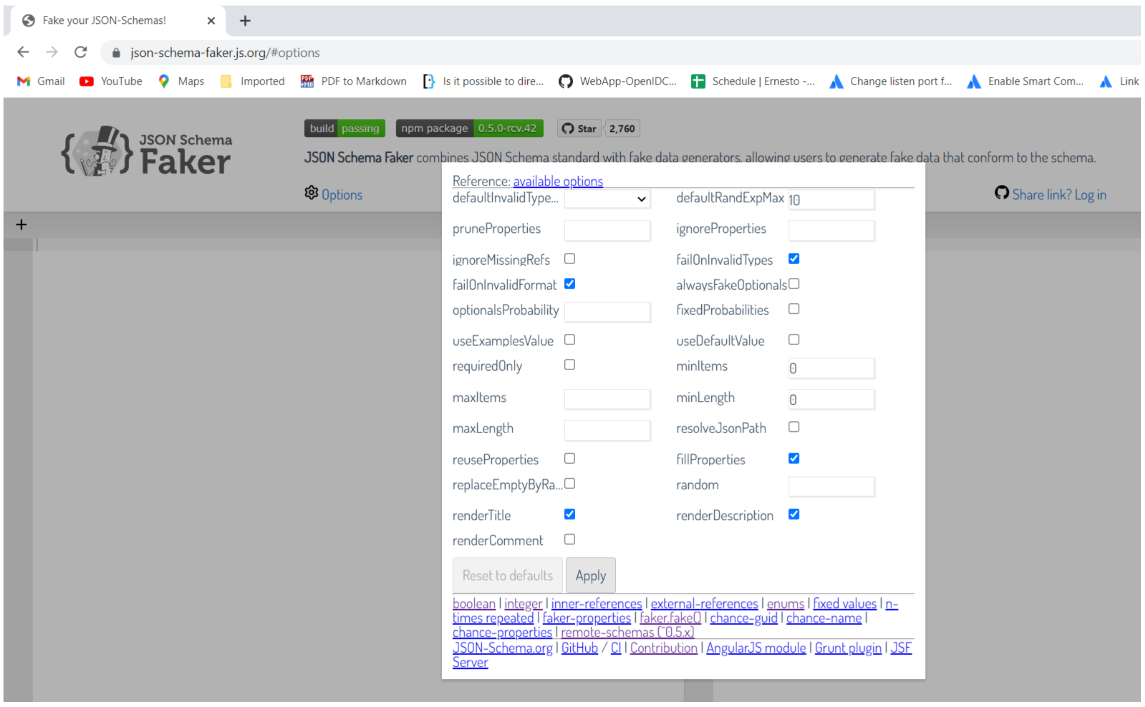
Boolean
true
false

Buzzword
contextually-based
radical
proactive

Can't find what you're looking for? You can mock almost anything using regular expressions...

JSON Schema Faker

Open http://json-schema-faker.js.org/ in browser and click **Options** link to get follow menu:



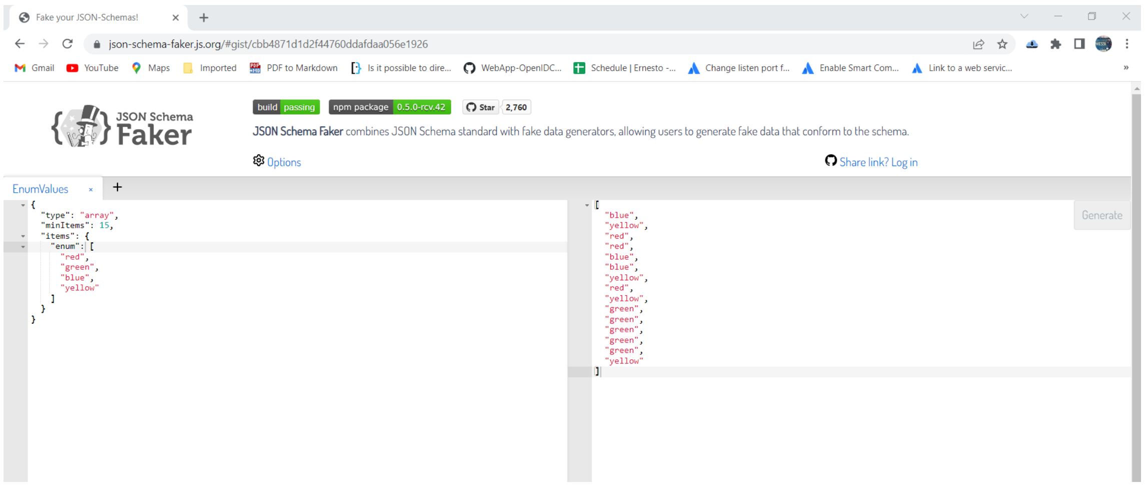
JSON Schema Faker combines JSON Schema standard with fake data generators, allowing users to generate fake data that conform to the schema.

Reference: available options

defaultInvalidType...		defaultRandExpMax	10
pruneProperties		ignoreProperties	
ignoreMissingRefs	☐	failOnInvalidTypes	✓
failOnInvalidFormat	✓	alwaysFakeOptionals	☐
optionalsProbability		fixedProbabilities	☐
useExamplesValue	☐	useDefaultValue	☐
requiredOnly	☐	minItems	0
maxItems		minLength	0
maxLength		resolveJsonPath	☐
reuseProperties	☐	fillProperties	✓
replaceEmptyByRa...	☐	random	
renderTitle	✓	renderDescription	✓
renderComment	☐		

boolean | integer | inner-references | external-references | enums | fixed values | n-times repeated | faker-properties | faker.fake() | chance-guid | chance-name | chance-properties | remote-schemas (0.5.x)
JSON-Schema.org | GitHub / CI | Contribution | AngularJS module | Grunt plugin | JSF Server

Click **Enum** from above menu and it will take you following screen. Click Generate to generate test data:



```
{
  "type": "array",
  "minItems": 15,
  "items": {
    "enum": [
      "red",
      "green",
      "blue",
      "yellow"
    ]
  }
}
```

```
[
  "blue",
  "yellow",
  "red",
  "red",
  "blue",
  "blue",
  "yellow",
  "red",
  "yellow",
  "green",
  "green",
  "green",
  "green",
  "green",
  "yellow"
]
```

Task:

1. Change minItems to `10` and change items names in enum and click Generate button again.
2. Click Options links and try different options i-e; **boolean** and **integer** etc.

JS Faker

This provides a command-line interface (CLI) to generate the data type based on your options, such as system, name, address, and phone. It can also output based on the specified locale language. It has been installed already: https://github.com/lestoni/faker-cli

Open terminal and try following examples:

```
faker-cli --helpers userCard

faker-cli --random uuid

faker-cli --locale de --helpers userCard
```