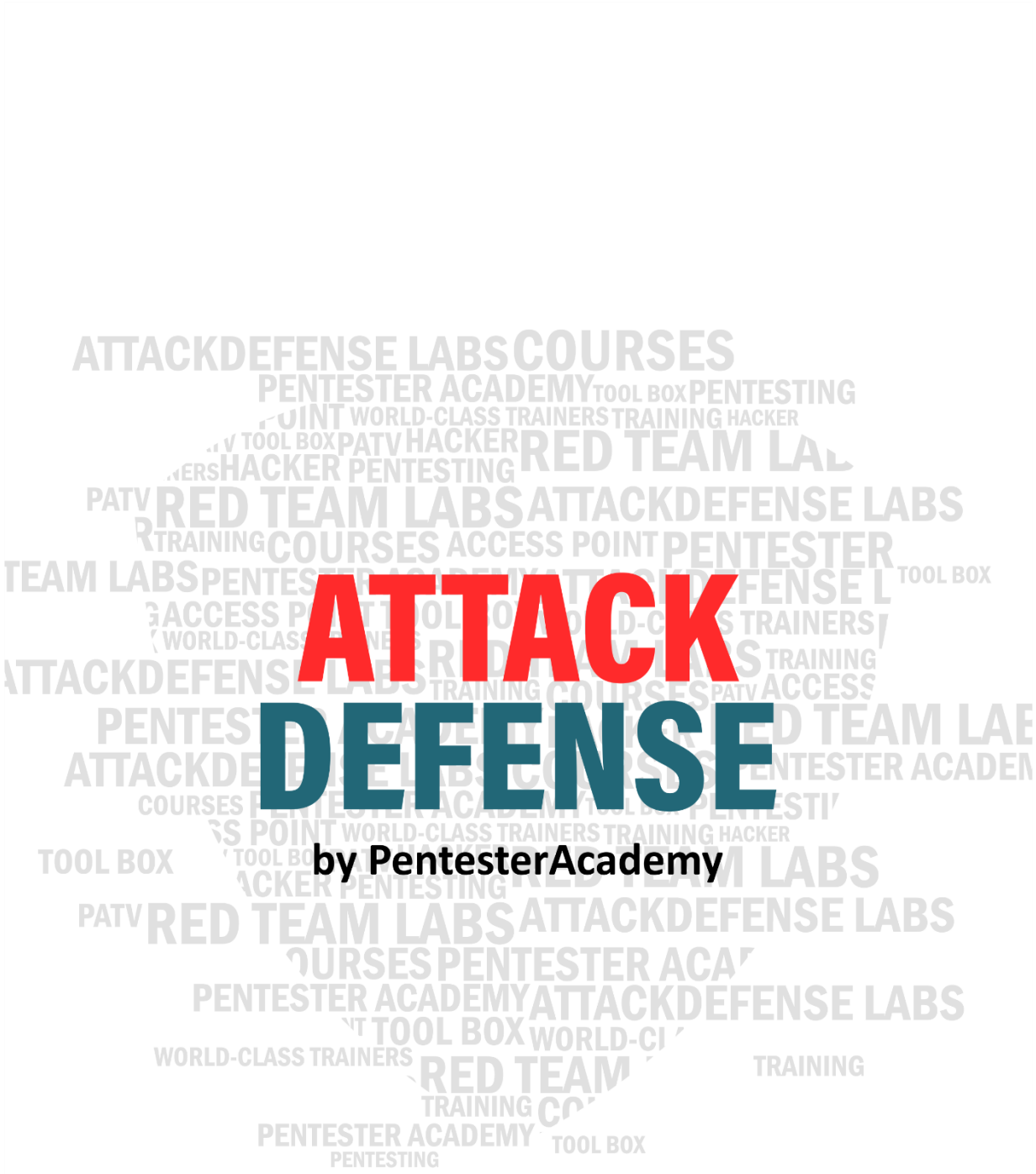
ATTACK
DEFENSE
by PentesterAcademy

Name	Pytest: Python Testing Framework
URL	https://attackdefense.com/challengedetails?cid=2260
Type	DevOps Basics: Functional Testing

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Challenge Description

Pytest is a testing based framework used to create automated test cases in python.

A Kali CLI machine (kali-cli) is provided to the user with pytest installed on it. The source code for three sample web applications is provided in the home directory of the root user.

Objective: Learn to write and run the tests using pytest framework!

Instructions:

- The source code of web applications is provided at /root/github-repos

Solution

Step 1: Check the provided web applications.

Command: ls -l github-repos/

```
root@attackdefense:~# ls -l github-repos/
total 12
drwxr-xr-x 11 root root 4096 Sep 13 09:49 coveralls-python
drwxr-xr-x  7 root root 4096 Sep 13 09:49 python3-project-template
drwxrwxr-x  5 root root 4096 Sep 13 09:49 python-serverless-boilerplate
root@attackdefense:~#
```

We will take one example at a time and run the tool on that and an example to create a custom test case.

Example 1: Coveralls Python

Step 1: Change into the coveralls-python directory

Commands:
cd github-repos/coveralls-python
ls

```
root@attackdefense:~# cd github-repos/coveralls-python
root@attackdefense:~/github-repos/coveralls-python#
root@attackdefense:~/github-repos/coveralls-python# ls
CHANGELOG.md  coveralls.egg-info  example      MANIFEST.in  README.rst  setup.py  tox.ini
coveralls     docs                LICENSE.txt  nonunicode   setup.cfg   tests
root@attackdefense:~/github-repos/coveralls-python#
```

Pytest tool will run all files of format **test_*.py** or ***_test.py** in the current directory and subdirectories. Although any filename can be specified explicitly.

Step 2: Change to the tests folder and list all the test files present in the codebase.

Commands:
cd tests
ls
ls api

```
root@attackdefense:~/github-repos/coveralls-python# cd tests
root@attackdefense:~/github-repos/coveralls-python/tests# ls
api  api_test.py  cli_test.py  conftest.py  git_test.py  __init__.py  __pycache__
root@attackdefense:~/github-repos/coveralls-python/tests# ls api
configuration_test.py  exception_test.py  __pycache__       wear_test.py
encoding_test.py       __init__.py        reporter_test.py
root@attackdefense:~/github-repos/coveralls-python/tests#
```

All those functions will be executed by Pytest whose name starts with **test**.

Step 3: Check the content of api_test.py.

Command: cat api_test.py

```
root@attackdefense:~/github-repos/coveralls-python/tests# cat api_test.py
import json
import os

import mock

from coveralls import Coveralls


@mock.patch.dict(os.environ, {}, clear=True)
def test_output_to_file(tmpdir):
    """Check we can write coveralls report into the file."""
    test_log = tmpdir.join('test.log')
    Coveralls(repo_token='xxx').save_report(test_log.strpath)
    report = test_log.read()

    assert json.loads(report)['repo_token'] == 'xxx'
root@attackdefense:~/github-repos/coveralls-python/tests#
```

The test will try to create a log file in the temp directory and will write ‘xxx’ in the file. After that, an attempt is made to read the temp file and its content. If the content matched with ‘xxx’ then the test is passed.

Step 5: Run the pytest command to test the codebase.

Commands:
cd ..
pytest

```
root@attackdefense:~/github-repos/coveralls-python/tests# cd ..
root@attackdefense:~/github-repos/coveralls-python# pytest
========================================= test session starts ==========================================
platform linux -- Python 3.8.5, pytest-5.1.0, py-1.9.0, pluggy-0.13.1
rootdir: /root/github-repos/coveralls-python
collected 52 items

tests/api_test.py .                                                                          [  1%]
tests/cli_test.py ..........                                                                 [ 21%]
tests/git_test.py .......                                                                    [ 34%]
tests/api/configuration_test.py .s............                                               [ 63%]
tests/api/encoding_test.py ..                                                                [ 67%]
tests/api/exception_test.py ...                                                              [ 73%]
tests/api/reporter_test.py ....                                                              [ 80%]
tests/api/wear_test.py ..........                                                            [100%]
```

```
========================================= warnings summary =========================================
/usr/lib/python3/dist-packages/requests/__init__.py:91
  /usr/lib/python3/dist-packages/requests/__init__.py:91: RequestsDependencyWarning: urllib3 (1.25.10) or cha
rdet (3.0.4) doesn't match a supported version!
    RequestsDependencyWarning)

/usr/lib/python3/dist-packages/socks.py:58
  /usr/lib/python3/dist-packages/socks.py:58: DeprecationWarning: Using or importing the ABCs from 'collectio
ns' instead of from 'collections.abc' is deprecated since Python 3.3,and in 3.9 it will stop working
    from collections import Callable

tests/api/configuration_test.py::Configuration::test_local_with_config
  /usr/local/lib/python3.7/dist-packages/yaml/constructor.py:126: DeprecationWarning: Using or importing the
ABCs from 'collections' instead of from 'collections.abc' is deprecated since Python 3.3,and in 3.9 it will s
top working
    if not isinstance(key, collections.Hashable):

-- Docs: https://docs.pytest.org/en/latest/warnings.html
================================== 51 passed, 1 skipped, 3 warnings in 1.97s ===================================
root@attackdefense:~/github-repos/coveralls-python#
```

Warnings:

- Deprecation Warning of packages
- Urllib3 doesn't match the supported version

The pytest successfully completed the tests with a detailed report for warnings.

Example 2: Python3 Project Template

Step 1: Change to the python3-project-template directory and check its contents.

Commands:
cd python3-project-template/
ls

```
root@attackdefense:~/github-repos# cd python3-project-template/
root@attackdefense:~/github-repos/python3-project-template#
root@attackdefense:~/github-repos/python3-project-template# ls
LICENSE  myapp  pytest.ini  README.md  requirements.txt  tests  tox.ini
root@attackdefense:~/github-repos/python3-project-template#
```

Step 2: Change to the tests folder and list all the test files present in the codebase.

Commands:
cd tests/
ls

```
root@attackdefense:~/github-repos/python3-project-template# cd tests/
root@attackdefense:~/github-repos/python3-project-template/tests#
root@attackdefense:~/github-repos/python3-project-template/tests# ls
__init__.py  __pycache__  test_damager.py
root@attackdefense:~/github-repos/python3-project-template/tests#
```

In this example, the Pytest tool will load the configurations from a “pytest.ini” file in the project directory and execute the instructions from test_damager.py

Step 3: Check the contents of “test_damager.py”.

Command: cat test_damager.py

```
root@attackdefense:~/github-repos/python3-project-template/tests# cat test_damager.py
import pytest
from myapp.damager import random_damage
```

```
from unittest import mock


input_data = [
    (dict(username='asasdf', email='bajgli+3@gmail.com', password='supersecret'), 200),
    (dict(username='goodusername', email='bajgli+4@gmail.com', password='asdfasdflong'), 200),
    (dict(username='badaboooo', email='bajgli+5@gmail.com', password='goodenough'), 200)
]


@pytest.mark.parametrize("test_input, expected_output", input_data)
def test_with_data(test_input, expected_output):
    assert test_input is not None
    assert 200 == expected_output
```

```
@pytest.mark.parametrize("test_input, expected_output", input_data)
def test_with_data(test_input, expected_output):
    assert test_input is not None
    assert 200 == expected_output


def test_a():
    assert True


@mock.patch("myapp.damager.randint", return_value=5, autospec=True)
def test_random_damage(mock_randint):
    assert random_damage(1) == 6
    mock_randint.assert_called_once_with(1, 8)
root@attackdefense:~/github-repos/python3-project-template/tests#
```

This test will only succeed when the test_input will have 200 as the value.

Step 4: Run the following command to check the content present in the “pytest.ini” file.

Commands:
cd ..
cat pytest.ini

```
root@attackdefense:~/github-repos/python3-project-template/tests# cd ..
root@attackdefense:~/github-repos/python3-project-template#
root@attackdefense:~/github-repos/python3-project-template# cat pytest.ini
[pytest]
addopts= --showlocals
python_paths= .:myapp
xfail_strict=true
log_format = %(levelname)8s  %(asctime)s  %(filename)20s  %(message)s
log_date_format = %m-%d %H:%M:%S
filterwarnings =
    error
    ignore::DeprecationWarning
root@attackdefense:~/github-repos/python3-project-template#
```

The file contains a pytest block of instructions. Following is the explanation of each instruction one by one.

addopts

This will add specified options to the command line argument. For example here the command will look like, *pytest --showlocals*

python_paths

This is a pytest plugin to insert the specified paths to the beginning of the PYTHONPATH environment variable before any tests run.

xfail_strict

Setting this to *true* means tests marked with *@pytest.mark.xfail* that actually succeed will by default fail the test suite.

log_format

Sets the format for logging messages.

log_date_format

Specifies the date format for logging messages.

filterwarnings

This is used to filter the warnings according to the listed rules. For example, here pytest will ignore deprecation warnings and turn all other warnings into errors.

Step 5: Run the pytest command to test the code of the repository.

Command: pytest

```
root@attackdefense:~/github-repos/python3-project-template# pytest
============================================ test session starts ============================================
platform linux -- Python 3.7.5, pytest-5.1.0, py-1.9.0, pluggy-0.13.1
rootdir: /root/github-repos/python3-project-template, inifile: pytest.ini
collected 5 items

tests/test_damager.py .....                                                                          [100%]

============================================= 5 passed in 0.03s =============================================
root@attackdefense:~/github-repos/python3-project-template#
```

The pytest command successfully completed the test.

Example 3: python serverless boilerplate

Step 1: Change to the python-serverless-boilerplate and check its contents.

Commands:
cd python-serverless-boilerplate/
ls

```
root@attackdefense:~/github-repos# cd python-serverless-boilerplate/
root@attackdefense:~/github-repos/python-serverless-boilerplate#
root@attackdefense:~/github-repos/python-serverless-boilerplate# ls
apps    docker-compose.structure.yml  docker-compose.yml  LICENSE   package.json  requirements.txt  tests
docker  docker-compose.test.yml       Dockerfile          Makefile  README.md     serverless.yml    tox.ini
root@attackdefense:~/github-repos/python-serverless-boilerplate#
```

Step 2: Change to the tests folder and list all the test files present in the codebase.

Commands:
cd tests
ls

```
root@attackdefense:~/github-repos/python-serverless-boilerplate# cd tests/
root@attackdefense:~/github-repos/python-serverless-boilerplate/tests#
root@attackdefense:~/github-repos/python-serverless-boilerplate/tests# ls
__init__.py  requirements.txt  status_test.py
root@attackdefense:~/github-repos/python-serverless-boilerplate/tests#
```

Step 3: Check the contents of “status_test.py”.

Command: cat status_test.py

```
root@attackdefense:~/github-repos/python-serverless-boilerplate/tests# cat status_test.py
import json
import unittest

from apps.status import handler


class StatusTest(unittest.TestCase):
    def test_handler_ok(self):
        res = handler.endpoint({}, {})
        self.assertEqual(res['statusCode'], 200)
        self.assertEqual(json.loads(res['body']), {'message':  'Hello world!'})
root@attackdefense:~/github-repos/python-serverless-boilerplate/tests#
```

This test will only succeed if the response code is 200 and the message is ‘Hello world!’

Step 4: Run the pytest command to test the code of the repository.

Commands:
cd ..
pytest

```
root@attackdefense:~/github-repos/python-serverless-boilerplate/tests# cd ..
root@attackdefense:~/github-repos/python-serverless-boilerplate#
root@attackdefense:~/github-repos/python-serverless-boilerplate# pytest
============================================ test session starts ============================================
platform linux -- Python 3.8.5, pytest-5.1.0, py-1.9.0, pluggy-0.13.1
rootdir: /root/github-repos/python-serverless-boilerplate
collected 1 item

tests/status_test.py .                                                                              [100%]

============================================ 1 passed in 0.03s =============================================
root@attackdefense:~/github-repos/python-serverless-boilerplate#
```

The pytest successfully completed the test.

Example 4: Create a test case to check if the random value is between 3 to 10 generated by 'test_damage' function of 'python3-project-template' project

Step 1: Check the source code of 'damage.py' located inside the 'myapp' directory

Command: cat ~/github-repos/python3-project-template/myapp/damager.py

```
root@attackdefense:~# cat ~/github-repos/python3-project-template/myapp/damager.py
from random import randint


def random_damage(modifier):
    roll = randint(1, 8)
    return modifier + roll
root@attackdefense:~#
```

The function will generate a random integer between 1 to 8 and add it with the modifier value passed in the function.

Step 2: Create a test case to check if the value generated from 'random_damage' function is between 3 to 10 where the value of modifier is 2.

Commands:
cd ~/github-repos/python3-project-template/tests
vim test_damager.py

```
import pytest
from myapp.damager import random_damage
from unittest import mock

input_data = [
    (dict(username='asasdf', email='bajgli+3@gmail.com', password='supersecret'), 200),
    (dict(username='goodusername', email='bajgli+4@gmail.com', password='asdfasdflong'), 200),
    (dict(username='badaboooo', email='bajgli+5@gmail.com', password='goodenough'), 200)
]


@pytest.mark.parametrize("test_input, expected_output", input_data)
def test_with_data(test_input, expected_output):
    assert test_input is not None
    assert 200 == expected_output


def test_a():
    assert True
```

Step 3: Write the code or test case to check for the value generated from random_damage function.

```
def test_damage():
        outcome = random_damage(2)
        assert 3 <= outcome <= 10
```

Command: cat test_damager.py

```
root@attackdefense:~/github-repos/python3-project-template/tests# cat test_damager.py
import pytest
from myapp.damager import random_damage
from unittest import mock


input_data = [
    (dict(username='asasdf', email='bajgli+3@gmail.com', password='supersecret'), 200),
    (dict(username='goodusername', email='bajgli+4@gmail.com', password='asdfasdflong'), 200),
    (dict(username='badaboooo', email='bajgli+5@gmail.com', password='goodenough'), 200)
```

```
@pytest.mark.parametrize("test_input, expected_output", input_data)
def test_with_data(test_input, expected_output):
    assert test_input is not None
    assert 200 == expected_output


def test_a():
    assert True

def test_damage():
        outcome = random_damage(2)
        assert 3 <= outcome <= 10
```

Step 4: Navigate back to the project directory and run the pytest to execute the test cases.

Commands:
cd ..
pytest

```
root@attackdefense:~/github-repos/python3-project-template/tests# cd ..
root@attackdefense:~/github-repos/python3-project-template# pytest
========================================= test session starts ==========================================
platform linux -- Python 3.8.5, pytest-5.1.0, py-1.9.0, pluggy-0.13.1
rootdir: /root/github-repos/python3-project-template, inifile: pytest.ini
collected 6 items

tests/test_damager.py ......                                                                    [100%]

========================================== 6 passed in 0.04s ===========================================
root@attackdefense:~/github-repos/python3-project-template#
```

All the 6 test cases executed successfully.

Learnings

Perform Functional testing using Pytest.