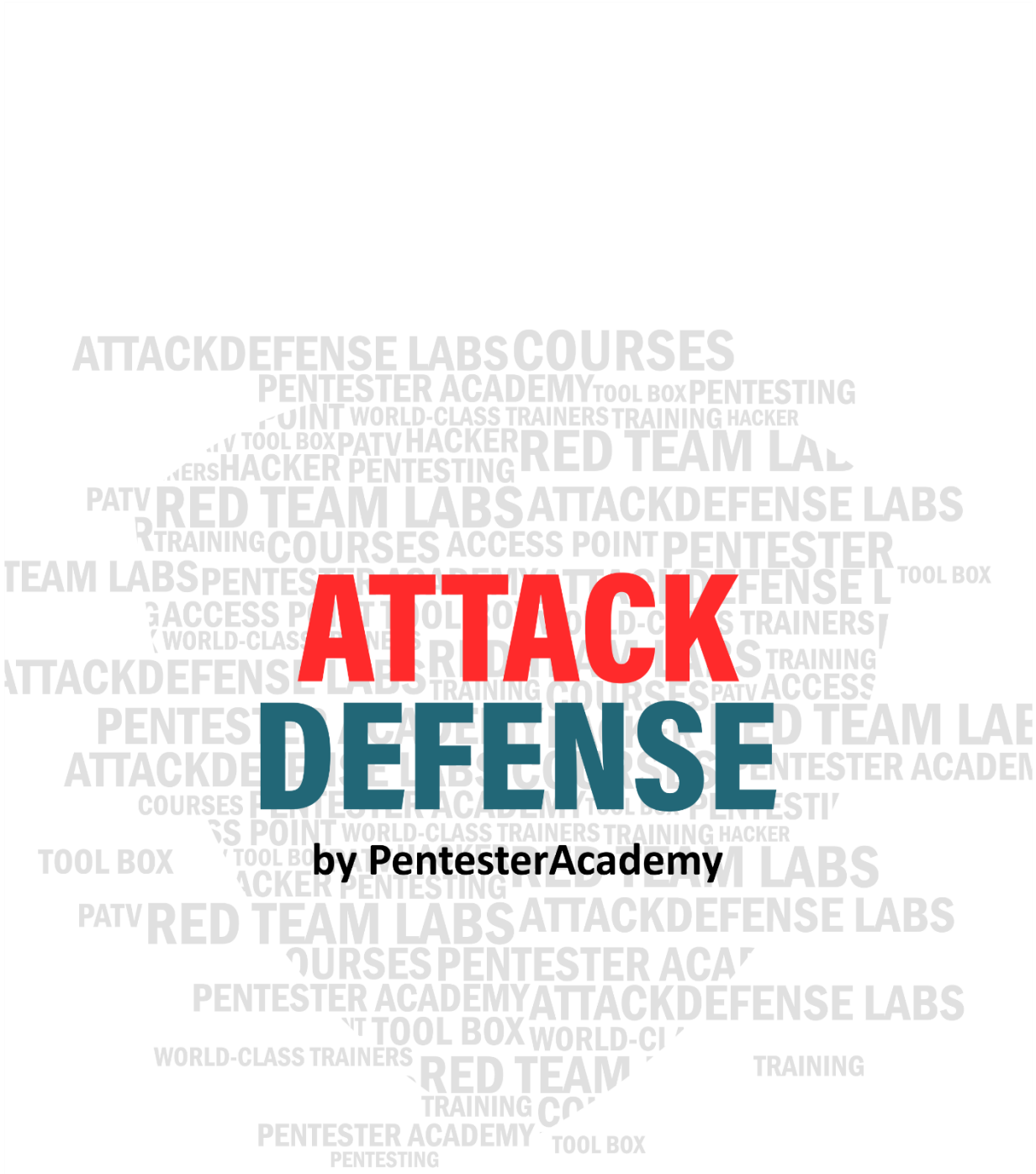
ATTACK
DEFENSE
by PentesterAcademy

Name	Tox: Multi-Environment Compatibility Testing
URL	https://attackdefense.com/challengedetails?cid=2255
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

Tox is a testing framework to test Python projects against different versions, packages and environments. This is to make sure that the project will run properly on different supported environments.

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

Objective: Utilise the tox to write and run tests to check web applications against different environments!

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 10 root root 4096 Sep 16 08:43 coveralls-python
drwxrwxr-x  6 root root 4096 Sep 16 08:56 python3-project-template
drwxr-xr-x  6 root root 4096 Sep 16 08:43 python-serverless-boilerplate
root@attackdefense:~#
```

Step 2: Check the available options of the tox tool.

Command: tox --help

```
root@attackdefense:~# tox --help
usage: tox [--version] [-h] [--help-ini] [-v] [-q] [--showconfig] [-l] [-a] [-c CONFIGFILE] [-e envlist] [--d
evenv ENVDIR] [--notest] [--sdistonly] [--skip-pkg-install] [-p [VAL]] [-o]
           [--parallel--safe-build] [--installpkg PATH] [--develop] [-i URL] [--pre] [-r] [--result-json PATH
] [--discover PATH [PATH ...]] [--hashseed SEED] [--force-dep REQ]
           [--sitepackages] [--alwayscopy] [-s [val]] [--workdir PATH]
           [args [args ...]]

tox options

positional arguments:
  args                          additional arguments available to command positional substitution (default
: None)
```

```
Environment variables
TOXENV: comma separated list of environments (overridable by '-e')
TOX_SKIP_ENV: regular expression to filter down from running tox environments
TOX_TESTENV_PASSENV: space-separated list of extra environment variables to be passed into test command envir
onments
PY_COLORS: 0 disable colorized output, 1 enable (default)
TOX_PARALLEL_NO_SPINNER: 1 disable spinner for CI, 0 enable (default)

root@attackdefense:~#
```

We will take one example at a time and run the tool on that.

Example 1: python3 project template

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

Commands:
cd ~/github-repos/python3-project-template
ls

```
root@attackdefense:~# cd github-repos/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#
```

The tox tool will look for “tox.ini” file in the project directory and execute the instructions present in it.

Step 2: Check the content present in the “tox.ini” file.

Command: cat tox.ini

```
root@attackdefense:~/github-repos/python3-project-template# cat tox.ini
[tox]
envlist=flake8,py38
skipsdist = true

[testenv]
basepython=python3.8
deps = -rrequirements.txt



[testenv:py37]
commands = py.test
root@attackdefense:~/github-repos/python3-project-template#
```

The file contains several blocks of instructions. Following is the explanation of each block one by one.

[tox]

This tests the project against the environments listed using ‘envlist’ that are flake8 and python3.8(py38). Flake8 and Python3.8 have to be installed in the environment otherwise tox will throw errors.

Setting ‘skipsdist’ to true will tell tox to not make a sdist, since the project doesn’t have a ‘setup.py’ file.

[testenv]

The ‘basepython’ argument specifies the python version to be used for a tox environment.

All the dependencies that are to be installed in the environment are specified by ‘deps’ argument. Here ‘requirements.txt’ file is specified which contains the dependencies.

[testenv:py37]

This is a python3.7(py37) specific test environment. Since python3.7(py37) is not specified in the environment list of the [tox] block, the commands specified using the ‘commands’ argument will not be executed.

Step 3: Run the tox command to test the code of the repository.

Command: tox

```
root@attackdefense:~/github-repos/python3-project-template# tox
/usr/local/lib/python3.8/dist-packages/tox/config/__init__.py:642: UserWarning: conflicting basepython versio
n (set 3.8, should be 3.7) for env 'py37';resolve conflict or set ignore_basepython_conflict
  warnings.warn(
flake8 run-test-pre: PYTHONHASHSEED='1239614451'
py38 installed: appdirs==1.4.4,atomicwrites==1.4.0,attrs==20.1.0,distlib==0.3.1,filelock==3.0.12,importlib-me
tadata==0.23,more-itertools==8.5.0,packaging==20.4,pluggy==0.13.1,py==1.9.0,pyparsing==2.4.7,pytest==5.1.0,si
x==1.15.0,toml==0.10.1,tox==3.13.2,virtualenv==20.0.31,wcwidth==0.2.5,zipp==3.1.0
py38 run-test-pre: PYTHONHASHSEED='1239614451'
_____________________________________________ summary _____________________________________________
  flake8: commands succeeded
  py38: commands succeeded
  congratulations :)
root@attackdefense:~/github-repos/python3-project-template#
```

Tox executed and successfully completed all of the tests.

Example 2: Coveralls Python

Step 1: Change to the coveralls-python directory and check its contents.

Commands:
cd ~/github-repos/coveralls-python/
ls

```
root@attackdefense:~/github-repos# cd 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#
```

Step 2: Run the following command to check the content present in the “tox.ini” file.

Command: cat tox.ini

```
root@attackdefense:~/github-repos/coveralls-python# cat tox.ini
[tox]
envlist = py38-cov41-{default,pyyaml},py{38}-cov{41,5}-{default,pyyaml}

[gh-actions]
python =
    3.8: py38-cov5

[testenv]
passenv = *
usedevelop = true
deps =
    mock
    responses
    pytest

[testenv:coveralls41]
deps =
    coverage>=4.1,<5.0
commands =
    coveralls --verbose

[testenv:coveralls5]
deps =
    coverage>=5.0,<6.0
commands =
    coveralls --verbose
root@attackdefense:~/github-repos/coveralls-python#
```

The file contains several blocks of instructions.

Following is the explanation of each block one by one.

[tox]

A generative envlist is specified with the following possibilities of environments:- py38-cov41-default, py38-cov41-pyyaml, py38-cov5-default, py38-cov5-pyyaml .

[gh-actions]

It's a tox plugin to run Github Actions. Here for Python3.8, it runs 'py38-cov5' environment.

[testenv]

'passenv' argument here is passing all the environment variables to the test environment.

Dependencies(mock, responses, pytest) are being installed using pip in development mode.

[testenv:coveralls41]

Dependencies and commands are specified for this environment.

[testenv:coveralls5]

Dependencies and commands are specified for this environment.

Step 3: Run the tox command to test the code of the repository.

Command: tox

```
root@attackdefense:~/github-repos/coveralls-python# tox
py38-cov41-default develop-inst-noop: /root/github-repos/coveralls-python
py38-cov41-default installed: attrs==20.1.0,certifi==2020.6.20,chardet==3.0.4,docopt==0.6.2,idna==2.10,inicon
fig==1.0.1,mock==4.0.2,more-itertools==8.5.0,packaging==20.4,pluggy==0.13.1,py==1.9.0,pyparsing==2.4.7,pytest
==6.0.1,requests==2.24.0,responses==0.12.0,six==1.15.0,toml==0.10.1,urllib3==1.25.10
py38-cov41-default run-test-pre: PYTHONHASHSEED='173157943'
py38-cov41-pyyaml develop-inst-noop: /root/github-repos/coveralls-python
py38-cov41-pyyaml installed: attrs==20.1.0,certifi==2020.6.20,chardet==3.0.4,docopt==0.6.2,idna==2.10,iniconf
ig==1.0.1,mock==4.0.2,more-itertools==8.5.0,packaging==20.4,pluggy==0.13.1,py==1.9.0,pyparsing==2.4.7,pytest=
=6.0.1,requests==2.24.0,responses==0.12.0,six==1.15.0,toml==0.10.1,urllib3==1.25.10
py38-cov41-pyyaml run-test-pre: PYTHONHASHSEED='173157943'
py38-cov5-default develop-inst-noop: /root/github-repos/coveralls-python
```

```
py38-cov5-default installed: attrs==20.1.0,certifi==2020.6.20,chardet==3.0.4,coverage==5.2.1,docopt==0.6.2,id
na==2.10,iniconfig==1.0.1,mock==4.0.2,more-itertools==8.5.0,packaging==20.4,pluggy==0.13.1,py==1.9.0,pyparsin
g==2.4.7,pytest==6.0.1,requests==2.24.0,responses==0.12.0,six==1.15.0,toml==0.10.1,urllib3==1.25.10
```

```
g==2.4.7,pytest==6.0.1,requests==2.24.0,responses==0.12.0,six==1.15.0,toml==0.10.1,urllib3==1.25.10
py38-cov5-default run-test-pre: PYTHONHASHSEED='173157943'
py38-cov5-pyyaml develop-inst-noop: /root/github-repos/coveralls-python
py38-cov5-pyyaml installed: attrs==20.1.0,certifi==2020.6.20,chardet==3.0.4,coverage==5.2.1,docopt==0.6.2,idn
a==2.10,iniconfig==1.0.1,mock==4.0.2,more-itertools==8.5.0,packaging==20.4,pluggy==0.13.1,py==1.9.0,pyparsing
==2.4.7,pytest==6.0.1,requests==2.24.0,responses==0.12.0,six==1.15.0,toml==0.10.1,urllib3==1.25.10
py38-cov5-pyyaml run-test-pre: PYTHONHASHSEED='173157943'
_____ summary _____
  py38-cov41-default: commands succeeded
  py38-cov41-pyyaml: commands succeeded
  py38-cov5-default: commands succeeded
  py38-cov5-pyyaml: commands succeeded
  congratulations :)
root@attackdefense:~/github-repos/coveralls-python#
```

Tox executed and successfully completed all of the tests.

Example 3: Python Serverless Boilerplate

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

Commands:
cd ~/github-repos/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: Run the following command to check the content present in the “tox.ini” file.

Command: cat tox.ini

```
root@attackdefense:~/github-repos/python-serverless-boilerplate# cat tox.ini
[tox]
envlist = py38,flake8,pylint
skipsdist = True
```

```
[testenv]
passenv = *
deps =
whitelist_externals = py.test
commands =
    py.test

[testenv:pylint]
setenv = VIRTUALENV_DIR = {envdir}
deps =
whitelist_externals = pylint
commands =
    pylint --rcfile=.pylintrc apps
```

```
[testenv:pylint]
setenv = VIRTUALENV_DIR = {envdir}
deps =
whitelist_externals = pylint
commands =
    pylint --rcfile=.pylintrc apps


[testenv:flake8]
deps =
whitelist_externals = flake8
commands =
    flake8 --config {toxinidir}/.flake8 apps
root@attackdefense:~/github-repos/python-serverless-boilerplate#
```

The file contains several blocks of instructions. Following is the explanation of each block one by one.

[tox]

Python3.8(py38), flake8 and pylint environments are tested.

Setting ‘skipsdist’ to true will tell tox to not make a sdist, since the project doesn’t have a ‘setup.py’ file.

[testenv]

'passenv' argument here is passing all the environment variables to the test environment. No dependencies are specified. 'whitelist_externals' specifies the command which can be executed in the 'commands' section without triggering a "not installed in virtualenv" warning. It is now deprecated, use 'allowlist_externals' instead.

py.test command is executed in the environment.

[testenv:pylint]

An environment variable 'VIRTUALENV_DIR' has been set to the current environment directory and 'pylint' command is executed.

[testenv:flake8]

'flake8' command is being executed.

Step 3: Run the tox command to test the code of the repository.

Command: tox

```
root@attackdefense:~/github-repos/python-serverless-boilerplate# tox
py38 recreate: /root/github-repos/python-serverless-boilerplate/.tox/py38
py38 run-test-pre: PYTHONHASHSEED='1596244235'
py38 run-test: commands[0] | py.test
============================================= test session starts ==============================================
platform linux -- Python 3.8.6, pytest-6.1.2, py-1.9.0, pluggy-0.13.1
cachedir: .tox/py38/.pytest_cache
rootdir: /root/github-repos/python-serverless-boilerplate
collected 1 item

tests/status_test.py .                                                                                  [100%]

============================================== 1 passed in 0.40s ===============================================
flake8 recreate: /root/github-repos/python-serverless-boilerplate/.tox/flake8
flake8 run-test-pre: PYTHONHASHSEED='1596244235'
flake8 run-test: commands[0] | flake8 --config /root/github-repos/python-serverless-boilerplate/.flake8 apps
pylint recreate: /root/github-repos/python-serverless-boilerplate/.tox/pylint
pylint run-test-pre: PYTHONHASHSEED='1596244235'
pylint run-test: commands[0] | pylint --rcfile=.pylintrc apps

-----------------------------------
Your code has been rated at 10.00/10
```

```
_____ summary _____
  py38: commands succeeded
  flake8: commands succeeded
  pylint: commands succeeded
  congratulations :)
root@attackdefense:~/github-repos/python-serverless-boilerplate#
```

Tox executed and successfully completed all of the tests.

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 open the tox.ini.

Commands:
cd ..
vim tox.ini

```
[tox]
envlist=flake8,py38
skipsdist = true

[testenv]
basepython=python3.8
deps = -rrequirements.txt


[testenv:py37]
commands = py.test
```

Step 5: Add the pytest instruction in the tox.ini under 'testenv'

```
root@attackdefense:~/github-repos/python3-project-template# cat tox.ini
[tox]
envlist=flake8,py38
skipsdist = true

[testenv]
basepython=python3.8
deps = -rrequirements.txt
commands= pytest


[testenv:py37]
commands = py.test
root@attackdefense:~/github-repos/python3-project-template#
```

Now when the tox starts running, it will execute the pytest command which will perform all the test cases defined.

Step 6: Run the tox command to trigger the tests

Command: tox

```
root@attackdefense:~/github-repos/python3-project-template# tox
/usr/local/lib/python3.8/dist-packages/tox/config/__init__.py:642: UserWarning: conflicting basepython versio
n (set 3.8, should be 3.7) for env 'py37';resolve conflict or set ignore_basepython_conflict
  warnings.warn(
flake8 installed: appdirs==1.4.4,atomicwrites==1.4.0,attrs==20.3.0,distlib==0.3.1,filelock==3.0.12,importlib-
metadata==0.23,more-itertools==8.6.0,packaging==20.4,pluggy==0.13.1,py==1.9.0,pyparsing==2.4.7,pytest==5.1.0,
six==1.15.0,toml==0.10.2,tox==3.13.2,virtualenv==20.1.0,wcwidth==0.2.5,zipp==3.4.0
flake8 run-test-pre: PYTHONHASHSEED='2136744801'
flake8 run-test: commands[0] | pytest
============================================ test session starts ============================================
platform linux -- Python 3.8.6, pytest-5.1.0, py-1.9.0, pluggy-0.13.1
cachedir: .tox/flake8/.pytest_cache
rootdir: /root/github-repos/python3-project-template, inifile: pytest.ini
collected 6 items

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

============================================= 6 passed in 0.03s =============================================
```

All the 6 test cases executed successfully.

Learnings

Perform functional testing with tox utility.