Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота №4

Ознайомлення з фреймворком pytest для тестування

Виконала

студентка групи ІТ-91:

Луцай Катерина

Перевірив:

Каплунов А. В.

Київ 2023

Мета: написати дві фікстури (для клієнта та сервера)..

Хід виконання роботи:

Запуск jn скрипту, окремих python файлів, та консольних команд відбувався на ОС Ubuntu.

```
[ ]  !pytest --version
     !iperf3 --version
     !netstat --version
```

```
pytest 7.2.2
iperf 3.9 (cJSON 1.7.13)
Linux k4tel 5.19.0-38-generic #39~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Fri Mar 17 21:16:15 UTC 2 x86_64
Optional features available: CPU affinity setting, IPv6 flow label, SCTP, TCP congestion algorithm setting,
net-tools 2.10-alpha
Fred Baumgarten, Alan Cox, Bernd Eckenfels, Phil Blundell, Tuan Hoang, Brian Micek and others
+NEW_ADDRT +RTF_IRTT +RTF_REJECT +FW_MASQUERADE +I18N +SELINUX
AF: (inet) +UNIX +INET +INET6 +IPX +AX25 +NETROM +X25 +ATALK +ECONET +ROSE -BLUETOOTH
HW:  +ETHER +ARC +SLIP +PPP +TUNNEL -TR +AX25 +NETROM +X25 +FR +ROSE +ASH +SIT +FDDI +HIPPI +HDLC/LAPB +EUI
```

```
[ ]  # show local ip adresses
     !ip --brief addr show
```

```
lo               UNKNOWN        127.0.0.1/8 ::1/128
enp0s31f6        DOWN
wlp4s0           UP             192.168.0.102/24 fe80::e76b:6cb0:42e9:5bd0/64
virbr0           DOWN           192.168.122.1/24
```

```
[ ]  import pytest
     from iperf_test import *
     from parser import *
     from conftest import *
```

```
[ ]  host_list, traffic_prompts, ssh_prompts
```

```
(['::1', '127.0.0.1', '192.168.0.102', '192.168.122.1'],
 [['iperf3', '-c', '192.168.0.102', '-p', '5201', '-t', '10'],
  ['iperf3', '-c', '192.168.0.102', '-p', '5201', '-t', '20'],
  ['iperf3', '-c', '192.168.0.102', '-p', '5201', '-t', '30'],
  ['iperf3', '-c', '192.168.0.102', '-p', '5201', '-t', '40'],
  ['iperf3', '-c', '192.168.0.102', '-p', '5201', '-t', '50']],
 ["netstat -tnlp | grep 5201 | awk '{print $4}'",
  "netstat -tnlp | grep iperf3 | awk '{print $4}'"])
```

```
# run tests and display output in verbose mode
pytest.main(["-v"])
```

```
============================= test session starts ==============================
platform linux -- Python 3.10.6, pytest-7.2.2, pluggy-1.0.0 -- /home/k/Documents/jn-env/jn-env/bin/python
cachedir: .pytest_cache
rootdir: /home/k/Documents/jn-env
plugins: anyio-3.6.2
collecting ... collected 40 items

iperf_test.py::TestSuite::test_ssh_client[::1] PASSED                    [  2%]
iperf_test.py::TestSuite::test_server[::1] PASSED                        [  5%]
iperf_test.py::TestSuite::test_ssh_servers[::1-netstat -tnlp | grep 5201 | awk '{print $4}'] PASSED [  7%]
iperf_test.py::TestSuite::test_ssh_servers[127.0.0.1-netstat -tnlp | grep 5201 | awk '{print $4}'] PASSED [ 10%]
iperf_test.py::TestSuite::test_ssh_client[127.0.0.1] PASSED              [ 12%]
iperf_test.py::TestSuite::test_server[127.0.0.1] PASSED                  [ 15%]
```

```
# run tests and display output in verbose mode
pytest.main(["-v"])
```

```
iperf_test.py::TestSuite::test_ssh_servers[127.0.0.1-netstat -tnlp | grep iperf3 | awk '{print $4}'] PASSED [ 17%]
iperf_test.py::TestSuite::test_ssh_servers[::1-netstat -tnlp | grep iperf3 | awk '{print $4}'] PASSED [ 20%]
iperf_test.py::TestSuite::test_ssh_servers[192.168.0.102-netstat -tnlp | grep iperf3 | awk '{print $4}'] PASSED [ 22%]
iperf_test.py::TestSuite::test_ssh_servers[192.168.0.102-netstat -tnlp | grep 5201 | awk '{print $4}'] PASSED [ 25%]
iperf_test.py::TestSuite::test_ssh_client[192.168.0.102] PASSED          [ 27%]
iperf_test.py::TestSuite::test_server[192.168.0.102] PASSED              [ 30%]
iperf_test.py::TestSuite::test_iperf_client_connection[192.168.0.102] PASSED [ 32%]
iperf_test.py::TestSuite::test_iperf_client_traffic[192.168.0.102-client0] PASSED [ 35%]
iperf_test.py::TestSuite::test_iperf_client_traffic[127.0.0.1-client0] PASSED [ 37%]
iperf_test.py::TestSuite::test_iperf_client_traffic[::1-client0] PASSED  [ 40%]
iperf_test.py::TestSuite::test_iperf_client_traffic[192.168.122.1-client0] PASSED [ 42%]
iperf_test.py::TestSuite::test_ssh_servers[192.168.122.1-netstat -tnlp | grep iperf3 | awk '{print $4}'] PASSED [ 45%]
iperf_test.py::TestSuite::test_ssh_servers[192.168.122.1-netstat -tnlp | grep 5201 | awk '{print $4}'] PASSED [ 47%]
iperf_test.py::TestSuite::test_ssh_client[192.168.122.1] PASSED          [ 50%]
iperf_test.py::TestSuite::test_server[192.168.122.1] PASSED              [ 52%]
iperf_test.py::TestSuite::test_iperf_client_connection[192.168.122.1] PASSED [ 55%]
iperf_test.py::TestSuite::test_iperf_client_traffic[192.168.122.1-client1] PASSED [ 57%]
iperf_test.py::TestSuite::test_iperf_client_traffic[192.168.0.102-client1] PASSED [ 60%]
iperf_test.py::TestSuite::test_iperf_client_traffic[127.0.0.1-client1] PASSED [ 62%]
iperf_test.py::TestSuite::test_iperf_client_traffic[::1-client1] PASSED  [ 65%]
iperf_test.py::TestSuite::test_iperf_client_traffic[192.168.122.1-client2] PASSED [ 67%]
iperf_test.py::TestSuite::test_iperf_client_traffic[192.168.0.102-client2] PASSED [ 70%]
iperf_test.py::TestSuite::test_iperf_client_traffic[127.0.0.1-client2] PASSED [ 72%]
iperf_test.py::TestSuite::test_iperf_client_traffic[::1-client2] PASSED  [ 75%]
iperf_test.py::TestSuite::test_iperf_client_traffic[192.168.122.1-client3] PASSED [ 77%]
iperf_test.py::TestSuite::test_iperf_client_traffic[192.168.0.102-client3] PASSED [ 80%]
iperf_test.py::TestSuite::test_iperf_client_traffic[127.0.0.1-client3] PASSED [ 82%]
iperf_test.py::TestSuite::test_iperf_client_traffic[::1-client3] FAILED  [ 85%]
iperf_test.py::TestSuite::test_iperf_client_traffic[192.168.122.1-client4] PASSED [ 87%]
iperf_test.py::TestSuite::test_iperf_client_traffic[192.168.0.102-client4] PASSED [ 90%]
iperf_test.py::TestSuite::test_iperf_client_traffic[127.0.0.1-client4] FAILED [ 92%]
iperf_test.py::TestSuite::test_iperf_client_traffic[::1-client4] PASSED  [ 95%]
iperf_test.py::TestSuite::test_iperf_client_connection[127.0.0.1] PASSED [ 97%]
iperf_test.py::TestSuite::test_iperf_client_connection[::1] PASSED       [100%]
```

```
================================== FAILURES ===================================
_____________ TestSuite.test_iperf_client_traffic[::1-client3] _____________

self = <iperf_test.TestSuite object at 0x7ffa17606ec0>
ssh_client = <paramiko.client.SSHClient object at 0x7ffa176ea770>
server = [None, None]
client = ('Connecting to host 192.168.0.102, port 5201\n[  5] local 192.168.0.102 port 44382 connected to 1

    def test_iperf_client_traffic(self, ssh_client, server, client):
        result, error, client_time = client

        assert error in ["", None]

        # standard parser headers in the output
        result_table = parser(result, client_time, parser_headers)
        condition_matches = result_table.loc[(result_table.Transfer > 2) & (result_table.Bitrate > 20), :]
        print(f"Percentage of matching conditions Transfer > 2 & Bitrate > 20:\t"
            f"{100 * len(condition_matches.index)/len(result_table.index)}%")
>       assert len(condition_matches.index) == len(result_table.index)
E       AssertionError

iperf_test.py:59: AssertionError
---------------------------- Captured stdout call -----------------------------
Headers of the output data: Interval, Transfer, Bitrate, Retr, Cwnd
Percentage of matching conditions Transfer > 2 & Bitrate > 20:  90.0%
```

```
_____________ TestSuite.test_iperf_client_traffic[127.0.0.1-client4] _____________

self = <iperf_test.TestSuite object at 0x7ffa17607220>
ssh_client = <paramiko.client.SSHClient object at 0x7ffa15d06020>
server = [None, None]
client = ('Connecting to host 192.168.0.102, port 5201\n[  5] local 192.168.0.102 port 57434 connected to 1

    def test_iperf_client_traffic(self, ssh_client, server, client):
        result, error, client_time = client

        assert error in ["", None]

        # standard parser headers in the output
        result_table = parser(result, client_time, parser_headers)
        condition_matches = result_table.loc[(result_table.Transfer > 2) & (result_table.Bitrate > 20), :]
        print(f"Percentage of matching conditions Transfer > 2 & Bitrate > 20:\t"
            f"{100 * len(condition_matches.index)/len(result_table.index)}%")
>       assert len(condition_matches.index) == len(result_table.index)
E       AssertionError

iperf_test.py:59: AssertionError
---------------------------- Captured stdout call -----------------------------
Headers of the output data: Interval, Transfer, Bitrate, Retr, Cwnd
Percentage of matching conditions Transfer > 2 & Bitrate > 20:  96.0%
========================== short test summary info ===========================
FAILED iperf_test.py::TestSuite::test_iperf_client_traffic[::1-client3] - AssertionError
FAILED iperf_test.py::TestSuite::test_iperf_client_traffic[127.0.0.1-client4] - AssertionError
================== 2 failed, 38 passed in 620.67s (0:10:20) ==================
<ExitCode.TESTS_FAILED: 1>
```

Провалені тести трафіку при замірах на 40 та 50 секундних інтервалах свідчать про 90% та 96% тестових замірів відповідних умовам перевірки на Transfer більше 2 Мбайт і пропускну здатність більше 20 Мбіт/с замість очікуваних 100%

parser.py

```
import re
import pandas as pd


def parser(output, samples=10, headers=None):
    # Define a regex pattern to match numerical values
    pattern = r'\d*\.\d+|\d+'
    result = []
    start_ind = 5

    for line in output.split('\n'):
        if not line:
            continue
        elif line.startswith('[ ID]'):
            if len(result) < samples:
                if headers is None:
                    start_ind = max(start_ind, line.rfind("]")) + 1
                    headers = [h.strip() for h in
line[start_ind:].split("  ") if h.strip() != ""]

                result = []
                print(f"Headers of the output data: {',
'.join(headers)}")
            else:
                break
        elif headers and line.startswith('[  '):
            # Extract all numerical values from the string using the
regex pattern
            num_matches = re.findall(pattern, line[start_ind:])
            if len(num_matches) < len(headers):
                continue
            else:
                # Convert the rest of the values to floats
                nums = [float(num) for num in num_matches]

                # Convert the first value to a float by subtracting two
integers
                nums[1] = abs(nums[1] - nums[0])
                # Slice only significant values
                nums = nums[1:]
                row_values = nums[:len(headers)] if len(nums) >
len(headers) else nums
                row_values[-1] = nums[-1]

                if len(result) < samples:
                    result.append(row_values)

        else:
            continue

    res_table = pd.DataFrame(result, columns=headers)
    return res_table
```

conftest.py

```
import subprocess
import paramiko
import pytest
from iperf_test import *

ssh_user = "kl"
ssh_pass = "1337"
ssh_port = 22
iperf_port = "5201"

local_ips_prompt = "ip addr show | grep -oP 'inet6? [^ /]+' | awk '{print
$2}' | sort | uniq | grep -v '^fe80::'"

# list of local host names
host_list = subprocess.getoutput(local_ips_prompt).split("\n")
host = host_list[2]

port_servers_prompt = "netstat -tnlp | grep " + iperf_port + " | awk
'{print $4}'"
iperf_servers_prompt = "netstat -tnlp | grep iperf3 | awk '{print $4}'"
# prompts to check server instance on the SSH client
ssh_prompts = [port_servers_prompt, iperf_servers_prompt]

# fast client coonection check flags
client_check_flags = ['-n', '1']
# iperf client base
client_start = ['iperf3', '-c', host, '-p', iperf_port]

# iperf server deamon start
server_start = "iperf3 -s -D"

# iperf servers stop
server_stop = f"echo {ssh_pass} | sudo -S killall iperf3"

# fast client connection check
check_prompt = ' '.join(client_start + client_check_flags)

# iperf client-server traffic measure prompts
traffic_prompts = []
for t in range(10, 60, 10):
    traffic_prompts.append(client_start + ["-t", str(t)])

# open SSH client connection
@pytest.fixture(scope="module", params=host_list)
def ssh_client(request):
    ssh_host = request.param or host

    ssh = paramiko.SSHClient()
    ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
```

```
    ssh.connect(ssh_host,
                port=ssh_port,
                username=ssh_user,
                password=ssh_pass)

    yield ssh
    ssh.close()

#check if the server is available for all local host client connections
@pytest.fixture(scope="module", params=ssh_prompts)
def ssh_prompt(ssh_client, server, request):
    prompt = request.param

    err = server[0]
    if err:
        return [err, None]

    stdin, stdout, stderr = ssh_client.exec_command(prompt)

    error = stderr.readlines()
    error = None if len(error) == 0 else [line.rstrip() for line in
error]

    output = stdout.readlines()
    output = None if len(output) == 0 else [line.rstrip() for line in
output]

    return [ error, output ]

# start iperf server on host SSH client
@pytest.fixture(scope="module")
def server(ssh_client):
    # start the server
    stdin, stdout, stderr = ssh_client.exec_command(server_start)

    error = stderr.readlines()
    error = None if len(error) == 0 else [line.rstrip() for line in
error]

    output = stdout.readlines()
    output = None if len(output) == 0 else [line.rstrip() for line in
output]

    yield [ error, output ]

    # stop the server
    stdin, stdout, stderr = ssh_client.exec_command(server_stop)


# connect iperf client to the server running on SSH host
@pytest.fixture(scope="module")
def client_check(server):
    err = server[0]
    if err:
```

```
        return None, err

    check_code, check_output = subprocess.getstatusoutput(check_prompt)
    return check_output, check_code

# test traffic of the iperf client to the server running on SSH host
@pytest.fixture(scope="module", params=traffic_prompts)
def client(server, request):
    err = server[0]
    if err:
        return None, err

    client_prompt = request.param
    process = subprocess.Popen(client_prompt,
                               stdout=subprocess.PIPE,
                               stderr=subprocess.PIPE,
                               encoding='utf-8')
    stdout, stderr = process.communicate()
    return stdout, stderr, int(client_prompt[-1])
```

iperf_test.py

```
import pytest
from parser import *
from conftest import *

parser_headers = ['Interval', 'Transfer', 'Bitrate', 'Retr', 'Cwnd']

class TestSuite():
    def test_ssh_client(self, ssh_client):
        ssh = ssh_client
        assert ssh.get_transport().is_active()

    def test_server(self, ssh_client, server):
        response = server
        assert response[0] in ["", None]

    def test_ssh_servers(self, ssh_client, ssh_prompt):
        response = ssh_prompt
        assert response[-1] not in ["", None]

        # forming dict of ports and IPs of running iperf instances
        iperf_instance_dict = {s.split(':')[-1]: s.split(':')[0] for s in
response[-1]}

        for port, ip in iperf_instance_dict.items():
            if len(ip) == 0: # not strict ip binding
                # check local ip addresses for available iperf client
connection on the current port
                iperf_instance_dict[port] = [ip for ip in host_list if

subprocess.getstatusoutput(f'iperf3 -c {ip} -p {port} -n 1')[0] == 0]
            assert len(iperf_instance_dict[port]) == len(host_list)
```

```
    def test_iperf_client_connection(self, ssh_client, server,
client_check):
        result, code = client_check
        assert code == 0

    def test_iperf_client_traffic(self, ssh_client, server, client):
        result, error, client_time = client

        assert error in ["", None]

        # standard parser headers in the output
        result_table = parser(result, client_time, parser_headers)
        condition_matches = result_table.loc[(result_table.Transfer > 2)
& (result_table.Bitrate > 20), :]
        print(f"Percentage of matching conditions Transfer > 2 & Bitrate
> 20:\t"
              f"{100 *
len(condition_matches.index)/len(result_table.index)}%")
        assert len(condition_matches.index) == len(result_table.index)
```

Код у git-hub: https://github.com/K4TEL/rt-sys.git

Висновки: було написано функціонал для підключення до ssh клієнта використовуючи бібліотеку paramiko, запуск сервера та підключення клієнтів до сервера використовуючи Python модуль subprocess, згенеровано мережевий трафік за допомогою iperf3, написано фікстури pytest для тестування отриманої системи клієнт-сервер в різних налаштуваннях трафіку, перевірки ssh клієнта на запуск сервера та підключення клієнтів з усіх локальних ip адрес.