**Network based attack for container: -**

**Pre-requisites before initiating the attack:**

This project requires installation of Docker, Scapy, pcapy, etc.., python libraries

**Docker Installation:** We need docker container to perform the attack. For this purpose, I am using Seed Ubuntu VM to install Docker.

Open new terminal and run following commands.

Let us update and upgrade the system first to install Docker, we could use following commands to do that.

**$ sudo apt update**

**$ sudo apt upgrade**

#Now let us install Docker and to do that we could use following command.

**$ sudo apt install docker.io**

Docker got installed.

Next, I have created two Docker file’s where I have written the commands of creating a docker image & ‘requirements.txt’ where python required library’s get stored, using text editor.

Docker file creation 1:

**FROM python:3.9-slim**

**WORKDIR /app**

**COPY . /app**

**RUN pip install --no-cache-dir -r requirements.txt**

**CMD ["python", "app.py"]**

Docker file creation 2:

**FROM python:3.9-slim**

**WORKDIR /monitoring**

**COPY monitoring.py /monitoring**

**RUN pip install psutil**

**CMD ["python", "monitoring.py"]**

Building the docker image:

**$ docker build -t my-python-app .**

**$ docker build -t mymonitoring .**

This command would build docker image named ‘my-python-app’ using current directory (‘.’) as a build context.

Runing Docker Container:

Open a terminal and use

**$ docker start mymonitoring**

**$ docker start my-python-app**

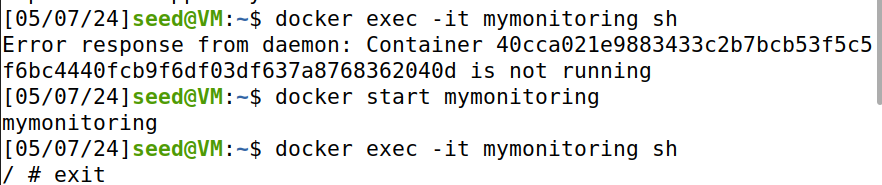
**$docker exec -it mymonitoring sh**

**/ #**

**$ docker exec -it my-python-app sh**

**/ #**

**Fig1:**

****

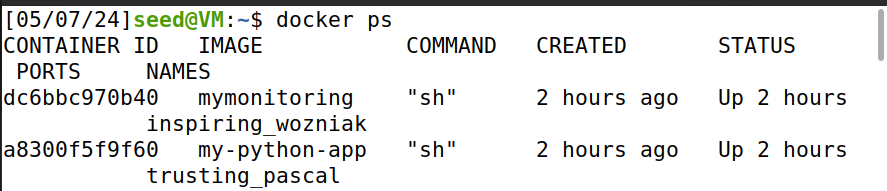
“/ #” indicates that we have entered their shell’s. enter ‘exit’ command to come out of the container shell.

To verify weather the containers are running?

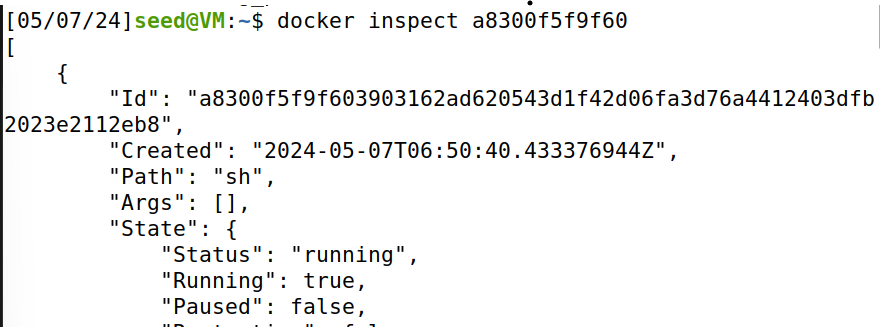
We could use this command **$ docker ps.** It’s output will show us the running containers.

Below screenshot shows us the successful running of both containers.

Fig 2:

****

**Fig3:**

****

**Fig 4:**

****

**Step-1: Performing SYN flood attack:**

For this purpose, open a new terminal and run the command

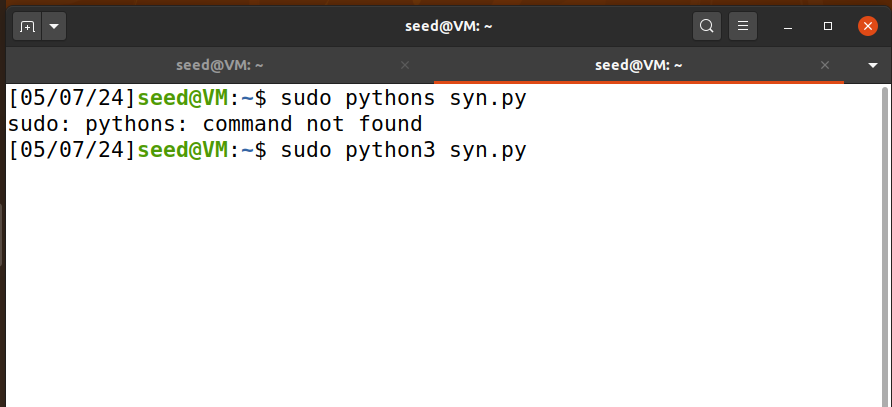
**$ sudo python3 syn.py**

The file syn.py contains the code to send SYN packets to targeted IP address.

Here our target IP address is ‘172.17.0.2’ (IP address of container ‘my-python-app’)

Below screenshot shows us the execution of syn.py command.

Fig 5:



**Step-2: Monitoring the attack:**

For this purpose, we are using another container called ‘mymonitoring’. In this container we will do the monitoring, detection and mitigation of SYN attack happening on ‘my-python-app’

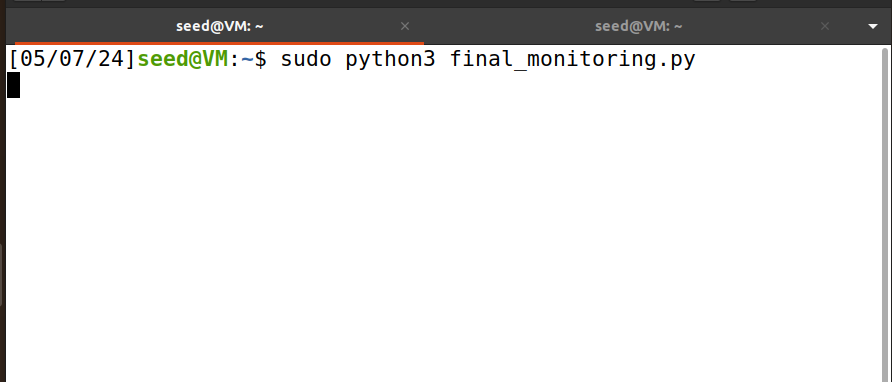
Open new terminal and verify weather the ‘mymonitoring’ container is running or not.

We have verified in the beginning, and the container is running.

Now run the command **$ sudo python3 final\_monitoring.py**

This command will start capturing the packets and store it in a file in the directory called ‘syn\_packets.pcap’.

Fig 6:



**Step-3: Detection and mitigation:**

Now the command **$ sudo python3 detect\_syn\_flood.py** would observe the packets capture in the ‘syn\_packets.pcap’ file and tally them, if the packets exceed the threshold indicated in the program file ‘detect\_syn\_flood.py’ then it would detect the attack and stop it.

Else it would print “No SYN flood attack detected. SYN count: x” (x could be any integer)

Since the syn attack is active, when we ran detection code, it blocked the incoming traffic from ip ‘ 172.17.0.1’ that the IP of my directory, we can observe it on fig 4.

I have used ‘subprocess’ library from python to block the block the IP address.

**subprocess.run(["iptables", "-A", "INPUT", "-s", ip\_address, "-j", "DROP"])**

**Fig 7:**

