Concordia Institute for Information System Engineering

Concordia University

INSE 6620: Cloud Computing Security and Privacy

Submitted to:

Professor Lingyu Wang

Term:

Summer 2023

Submitted By:

NAME	STUDENT ID
Mandeep Kaur	40219089
Kamal Sharma	40219383
Himani Rajput	40167840
Suman Bajwa	40217710
Darshit Gajjar	40224405
Aditya Raj Arora	40217177
Rushi Ashwinbhai Chandalia	40231722
Abhay Abraham Kottackattu	40220896

TABLE OF CONTRIBUTION:

Installation And Setup	Rushi, Darshit, Abhay
Attacks and Defense	Mandeep, Kamal, Rushi, Darshit
Report	Suman, Himani, Aditya, Kamal
Presentation	Mandeep, Abhay, Aditya

TABLE OF CONTENTS

INTRODUCTION

In today's rapidly evolving technological landscape, the security and stability of computer networks are of paramount importance. The ability to create, manage, and safeguard virtual networks is a crucial skill for network administrators and cybersecurity professionals. This project revolves around the installation and configuration of OpenStack, a powerful open-source cloud computing platform, to establish a virtual network environment. Additionally, the project involves implementing and detecting a network-based attack within this virtual network, highlighting the significance of robust security measures. [11]

The project is divided into two distinct phases: the installation and deployment of OpenStack for setting up a virtual network, and the execution and detection of a network-based attack to emphasize the importance of network security protocols.

INSTALLATION

To install OpenStack on Ubuntu, certain prerequisites must be met, including Ubuntu OS, a minimum of 4 GB RAM, a multi-core enabled processor, at least 10 GB of free hard disk space, and a stable internet connection.

We followed the following steps to install OpenStack using Devstack: [1]

STEP 1: System Preparation

To prepare the system, we started by updating the system using the command:

sudo apt-get update && sudo apt-get upgrade -y

After running the command successfully, it will prompt to ask for root privileges. The system will start upgrading after entering the password.

STEP 2: **Creating Stack user with sudo privilege**

Using the following command, we created a new user named "stack" to install OpenStack and enable the stack user to have root privileges.

sudo useradd -s /bin/bash -d /opt/stack -m stack

echo "stack ALL=(ALL) NOPASSWD: ALL" | sudo tee /etc/sudoers.d/stack

After the creation of stack user login using the command:

sudo su – stack

STEP 3: **Downloading Devstack**

Install Git and then clone devstack from its repository using the command:

git clone https://opendev.org/openstack/devstack [7]

STEP 4: Creating Configuration File

Navigate to the Devstack folder and create a local.conf configuration file:

cd devstack

vim local.conf

Paste the following configuration into the file: [5]

[[local|localrc]]

ADMIN_PASSWORD=StrongAdminSecret

DATABASE_PASSWORD=$ADMIN_PASSWOCinder

RABBIT_PASSWORD=$ADMIN_PASSWORD

SERVICE_PASSWORD=$ADMIN_PASSWORD

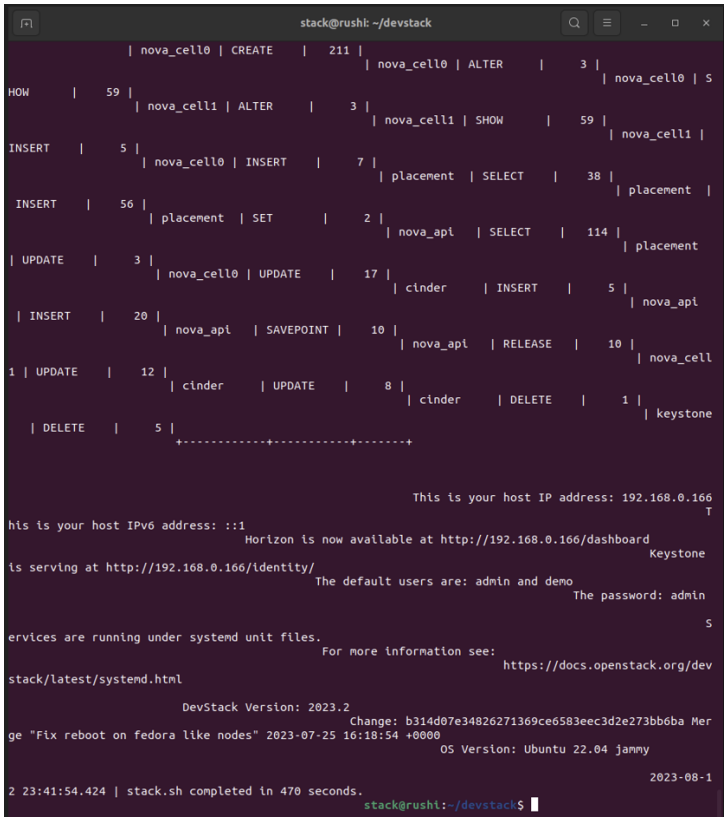
Save and exit.

STEP 5: Installing Openstack

Run the setup script to install OpenStack:

./stack.sh

The script will install various components including Horizon, Keystone, Nova Glance, Neutron, Placement, and Cinder.



STEP 6: Accessing OpenStack via Web Browser

After the installation completes, access the OpenStack Dashboard through your web browser using the URL:

https://server-ip/dashboard

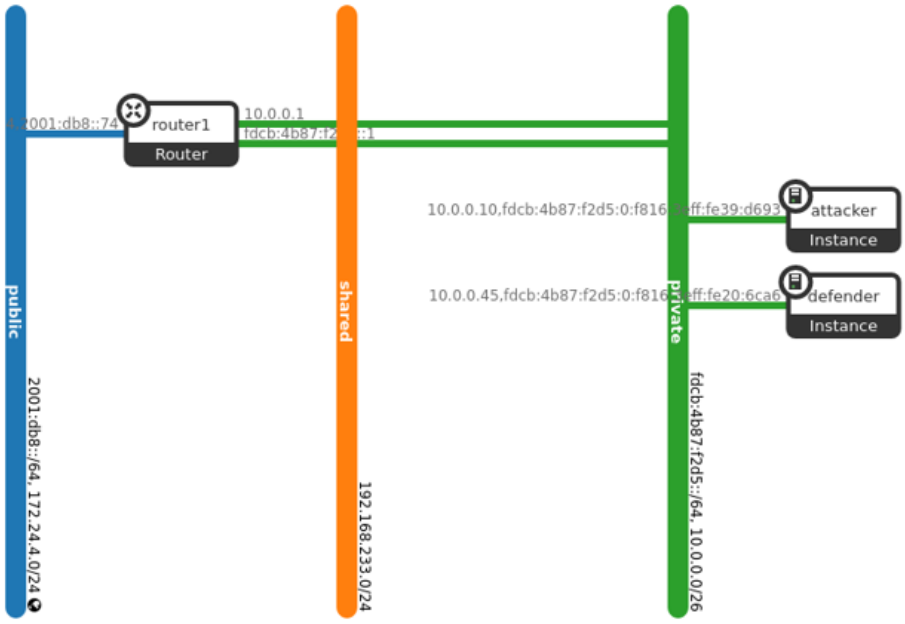
It will open the OpenStack login page as shown in the image below:

SETUP AND NETWORK TOPOLOGY:

For this project. To perform Attack and Detection 2 Ubuntu instances (Attacker and Defender) are created on a network named Private. This Private network is connected to a Router using one interface and another interface is used to connect to the public network which is Internet. The IP addresses of both Ubuntu instances and screenshot for the network topology is given below.

Attacker IP: 10.0.0.10

Defender IP: 10.0.0.45



Configuration used to launch the instances:

Source: Ubuntu

Flavor: de512M (VCPUS: 1, RAM: 512 MB, DISK: 5 GB)

Network: Private

Security Groups: (SSH, ICMP) merged with Default security group

Key Pair: SSH Public Key (Generated using “SSH keygen –b 4096” command)

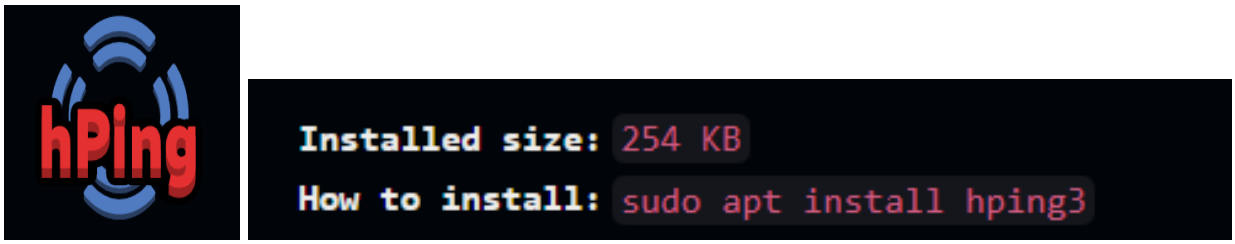
ATTACKS AND DETECTION:

In this project we performed three network attacks (2 active and 1 passive) and tried to detect these using an open-source Intrusion Prevention System named snort. All three attack scenarios and the subsequent response by the defender are highlighted in the sections that follow.

ICMP FLOOD - Attack

In the first scenario, the attacker instance tries to perform an ICMP (Internet Control Message Protocol) flood attack on the network. While the protocol is not inherently vulnerable by design it does facilitate the performance of several lethal network attacks due to its rather simple nature as can be seen in this scenario. In this particular case the attacker utilizes the ping capability of the ICMP protocol to flood the

network with millions of ICMP packets. This is achieved through the use of an open-source tool called hping3.

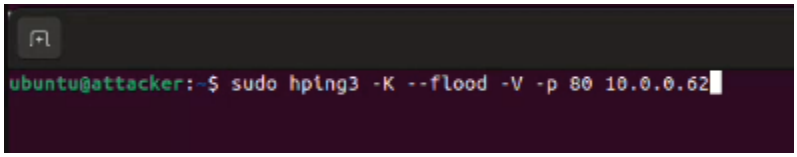


Hping is a network tool that can send custom TCP/UDP/ICMP packets in order to display target replies just like ICMP. It was developed by Savaltore Sanfillipo in 2006 as a packet analyzer/assembler that can also be used to send files using a covered channel

SYNTAX

Hping3 host [options] [2]

For this project we utilized hping3 as listed in screenshot 1



Screenshot 1

The modifiers used are:

-K specify icmp code (default is 0) this packet specifies that the destination network is unreachable

--flood send packets as fast as possible. This option does not show the replies from the host

-V verbose mode

-p specify port number

```
ubuntu@attacker:~$ sudo hping3 -K --flood -V -p 80 10.0.0.62
sudo: unable to resolve host attacker: Name or service not known
using ens3, addr: 10.0.0.25, MTU: 1442
HPING 10.0.0.62 (ens3 10.0.0.62): icmp mode set, 28 headers + 0 data bytes
len=28 ip=10.0.0.62 ttl=64 id=43282 tos=0 iplen=28
icmp_seq=0 rtt=5.9 ms
len=28 ip=10.0.0.62 ttl=64 id=43413 tos=0 iplen=28
icmp_seq=1 rtt=5.7 ms
len=28 ip=10.0.0.62 ttl=64 id=43516 tos=0 iplen=28
icmp_seq=2 rtt=5.4 ms
len=28 ip=10.0.0.62 ttl=64 id=43544 tos=0 iplen=28
icmp_seq=3 rtt=5.1 ms
len=28 ip=10.0.0.62 ttl=64 id=43679 tos=0 iplen=28
icmp_seq=4 rtt=4.7 ms
len=28 ip=10.0.0.62 ttl=64 id=43908 tos=0 iplen=28
icmp_seq=5 rtt=4.5 ms
len=28 ip=10.0.0.62 ttl=64 id=43964 tos=0 iplen=28
icmp_seq=6 rtt=4.2 ms
```

Screenshot 1.1

ICMP FLOOD – Detection

For Detecting this attack, we utilized snort. Snort was developed in 1998 by Martin Roesch. It is now developed and maintained by Cisco which purchased Sourcefire, Martin's company, in 2013. Snort utilizes rules to help define malicious network activity. All packets encountered within the network are then matched against these rules by snort and when a matching packet is encountered an alert is raised.

For detecting the ICMP flood in this scenario we added the ICMP alert rule (as seen in screen shot 2) to the local.rules file which can be found at the following path

/etc/snort/rules/local.rules

```
ubuntu@defender:~$ sudo cat /etc/snort/rules/local.rules
sudo: unable to resolve host defender: Name or service not known
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# ----------------
# LOCAL RULES
alert icmp any any -> $HOME_NET any (msg:"ICMP flood"; sid:1000001; rev:1;)
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.
```

Screenshot 2

Once the attack commences Snort is successfully able to detect the attack as seen below in screenshot 3

```
ubuntu@defender:~$ sudo snort -A console -q -c /etc/snort/snort.conf -i ens3
sudo: unable to resolve host defender: Name or service not known
08/13-16:15:26.445836  [**] [1:1000001:1] ICMP flood [**] [Priority: 0] {ICMP} 10.0.0.25 -> 10.0.0.62
08/13-16:15:26.445882  [**] [1:1000001:1] ICMP flood [**] [Priority: 0] {ICMP} 10.0.0.62 -> 10.0.0.25
08/13-16:15:27.446577  [**] [1:1000001:1] ICMP flood [**] [Priority: 0] {ICMP} 10.0.0.25 -> 10.0.0.62
08/13-16:15:27.446619  [**] [1:1000001:1] ICMP flood [**] [Priority: 0] {ICMP} 10.0.0.62 -> 10.0.0.25
08/13-16:15:28.446254  [**] [1:1000001:1] ICMP flood [**] [Priority: 0] {ICMP} 10.0.0.25 -> 10.0.0.62
08/13-16:15:28.446302  [**] [1:1000001:1] ICMP flood [**] [Priority: 0] {ICMP} 10.0.0.62 -> 10.0.0.25
08/13-16:15:29.446540  [**] [1:1000001:1] ICMP flood [**] [Priority: 0] {ICMP} 10.0.0.25 -> 10.0.0.62
08/13-16:15:29.446580  [**] [1:1000001:1] ICMP flood [**] [Priority: 0] {ICMP} 10.0.0.62 -> 10.0.0.25
08/13-16:15:30.446897  [**] [1:1000001:1] ICMP flood [**] [Priority: 0] {ICMP} 10.0.0.25 -> 10.0.0.62
08/13-16:15:30.446937  [**] [1:1000001:1] ICMP flood [**] [Priority: 0] {ICMP} 10.0.0.62 -> 10.0.0.25
08/13-16:15:31.447138  [**] [1:1000001:1] ICMP flood [**] [Priority: 0] {ICMP} 10.0.0.25 -> 10.0.0.62
08/13-16:15:31.447180  [**] [1:1000001:1] ICMP flood [**] [Priority: 0] {ICMP} 10.0.0.62 -> 10.0.0.25
08/13-16:15:32.447434  [**] [1:1000001:1] ICMP flood [**] [Priority: 0] {ICMP} 10.0.0.25 -> 10.0.0.62
08/13-16:15:32.447478  [**] [1:1000001:1] ICMP flood [**] [Priority: 0] {ICMP} 10.0.0.62 -> 10.0.0.25
08/13-16:15:33.447730  [**] [1:1000001:1] ICMP flood [**] [Priority: 0] {ICMP} 10.0.0.25 -> 10.0.0.62
08/13-16:15:33.447772  [**] [1:1000001:1] ICMP flood [**] [Priority: 0] {ICMP} 10.0.0.62 -> 10.0.0.25
```

Screenshot 3

Nmap Scan – Attack

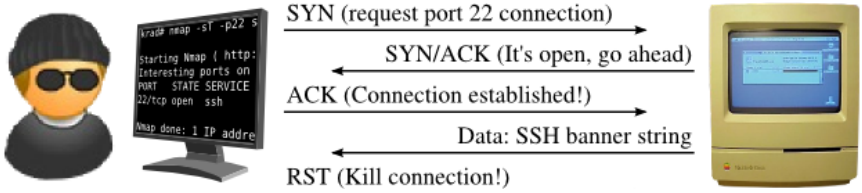
In the second scenario we attempt to perform a passive attack on the network where the attacker instance launches nmap to scan the victim machine. Nmap or Network Mapper is an open-source tool used for network discovery and security auditing.

```
test@test-virtual-machine:~$ sudo apt install nmap
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
```

SYNTAX

Nmap [options] host IP

Nmap works by sending specially crafted packets to target hosts and then identifies them by analyzing the response received. For this attack we use the –sT option of nmap which is the connect () scan. [9]



In this type of scan, Nmap issues a connect system call which is a high-level system call similar to that used by web browsers, P2P clients, and most other network applications, to establish a connection. This is part of a programming interface known as the Berkeley Sockets API. Nmap uses this API to obtain status information on each connection attempt.

```
ubuntu@attacker:~$ nmap -sT 10.0.0.62
Starting Nmap 7.80 ( https://nmap.org ) at 2023-08-13 16:23 UTC
Nmap scan report for 10.0.0.62
Host is up (0.00015s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
```

In our sample attack scenario, Nmap is able to ascertain that port 22 is open on the defender machine.

Nmap Scan – Detection

Just like the previous scenario, we will edit the local.rules file to add a tcp alert. This enables snort to raise an alert on the terminal as soon any tcp packets addressed to the defender's subnet are encountered.

```
ubuntu@defender:~$ sudo cat /etc/snort/rules/local.rules
sudo: unable to resolve host defender: Name or service not known
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# ----------------
# LOCAL RULES
alert icmp any any -> $HOME_NET any (msg:"ICMP flood"; sid:1000001; rev:1;)
alert tcp any any -> $HOME_NET any (msg:"NMAP TCP scan"; sid: 10000002; rev:2;)
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.
```

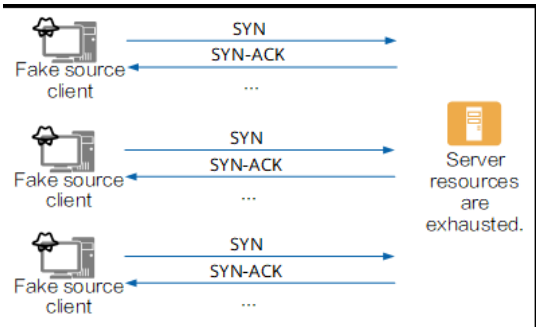
Once the scan commences from the attacker's machine, snort is able to alert the defender's terminal to the malicious activity going on in the network.

```
ubuntu@defender:~$ sudo snort -A console -q -c /etc/snort/snort.conf -i ens3
sudo: unable to resolve host defender: Name or service not known
```

```
08/13-16:23:42.263638  [**] [1:10000002:2] NMAP TCP scan [**] [Priority: 0] {TCP} 10.0.0.62:3918 -> 10.0.0.25:56510
08/13-16:23:42.263647  [**] [1:10000002:2] NMAP TCP scan [**] [Priority: 0] {TCP} 10.0.0.62:8291 -> 10.0.0.25:37498
08/13-16:23:42.263653  [**] [1:10000002:2] NMAP TCP scan [**] [Priority: 0] {TCP} 10.0.0.62:1031 -> 10.0.0.25:38662
08/13-16:23:42.263661  [**] [1:10000002:2] NMAP TCP scan [**] [Priority: 0] {TCP} 10.0.0.62:1022 -> 10.0.0.25:45458
08/13-16:23:42.263684  [**] [1:10000002:2] NMAP TCP scan [**] [Priority: 0] {TCP} 10.0.0.62:6881 -> 10.0.0.25:38880
08/13-16:23:42.263720  [**] [1:10000002:2] NMAP TCP scan [**] [Priority: 0] {TCP} 10.0.0.62:1533 -> 10.0.0.25:42648
08/13-16:23:42.263730  [**] [1:10000002:2] NMAP TCP scan [**] [Priority: 0] {TCP} 10.0.0.62:280 -> 10.0.0.25:35794
08/13-16:23:42.263745  [**] [1:10000002:2] NMAP TCP scan [**] [Priority: 0] {TCP} 10.0.0.62:18101 -> 10.0.0.25:54842
08/13-16:23:42.263763  [**] [1:10000002:2] NMAP TCP scan [**] [Priority: 0] {TCP} 10.0.0.62:417 -> 10.0.0.25:53934
08/13-16:23:42.263778  [**] [1:10000002:2] NMAP TCP scan [**] [Priority: 0] {TCP} 10.0.0.62:32782 -> 10.0.0.25:58200
08/13-16:23:42.263786  [**] [1:10000002:2] NMAP TCP scan [**] [Priority: 0] {TCP} 10.0.0.62:902 -> 10.0.0.25:33188
08/13-16:23:42.263802  [**] [1:10000002:2] NMAP TCP scan [**] [Priority: 0] {TCP} 10.0.0.62:9575 -> 10.0.0.25:35380
08/13-16:23:42.263823  [**] [1:10000002:2] NMAP TCP scan [**] [Priority: 0] {TCP} 10.0.0.62:42510 -> 10.0.0.25:54238
08/13-16:23:42.263858  [**] [1:10000002:2] NMAP TCP scan [**] [Priority: 0] {TCP} 10.0.0.62:8087 -> 10.0.0.25:47790
08/13-16:23:42.263873  [**] [1:10000002:2] NMAP TCP scan [**] [Priority: 0] {TCP} 10.0.0.62:34571 -> 10.0.0.25:33366
08/13-16:23:42.263878  [**] [1:10000002:2] NMAP TCP scan [**] [Priority: 0] {TCP} 10.0.0.62:800 -> 10.0.0.25:43858
08/13-16:23:42.263883  [**] [1:10000002:2] NMAP TCP scan [**] [Priority: 0] {TCP} 10.0.0.62:8042 -> 10.0.0.25:53958
08/13-16:23:42.263911  [**] [1:10000002:2] NMAP TCP scan [**] [Priority: 0] {TCP} 10.0.0.62:3828 -> 10.0.0.25:56376
```

SYN FLOOD – Attack

In the third scenario, we attempt to perform a syn flood attack on the network using hping. In its machination the syn flood attack is quite similar to the ICMP flood however the crucial difference lies in the packets used to conduct the actual attack. While the ICMP flood used regular ping packets to interact with the victim, SYN flood utilizes TCP syn packets to establish a partial handshake with the victim.



For conducting this attack using hping the following syntax was used:

```
ubuntu@attacker:~$ sudo hping3 -S --flood -V -p 80 10.0.0.62
sudo: unable to resolve host attacker: Temporary failure in name resolution
using ens3, addr: 10.0.0.25, MTU: 1442
HPING 10.0.0.62 (ens3 10.0.0.62): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 10.0.0.62 hping statistic ---
180889 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
```

SYN FLOOD – Detection

For detecting this attack, we needn't add any new rules to snort as our previous alert for detecting any malicious TCP packets would trigger an alert. This is because the SYN packet which is used for conducting the SYN flood attack is ultimately a TCP packet. Therefore, snort is able to flag it as an incident.

```
08/13-18:18:27.581108  [**] [1:10000002:2] TCP packets detected, POSSIBLE SYN FLOOD [**] [Priority: 0] {TCP} 10.0.0.62:80 -> 10.0.0.25:2490
08/13-18:18:27.581110  [**] [1:10000002:2] TCP packets detected, POSSIBLE SYN FLOOD [**] [Priority: 0] {TCP} 10.0.0.62:80 -> 10.0.0.25:2491
08/13-18:18:27.581112  [**] [1:10000002:2] TCP packets detected, POSSIBLE SYN FLOOD [**] [Priority: 0] {TCP} 10.0.0.62:80 -> 10.0.0.25:2492
08/13-18:18:27.581114  [**] [1:10000002:2] TCP packets detected, POSSIBLE SYN FLOOD [**] [Priority: 0] {TCP} 10.0.0.62:80 -> 10.0.0.25:2493
08/13-18:18:27.581116  [**] [1:10000002:2] TCP packets detected, POSSIBLE SYN FLOOD [**] [Priority: 0] {TCP} 10.0.0.62:80 -> 10.0.0.25:2494
08/13-18:18:27.581118  [**] [1:10000002:2] TCP packets detected, POSSIBLE SYN FLOOD [**] [Priority: 0] {TCP} 10.0.0.62:80 -> 10.0.0.25:2495
08/13-18:18:27.581120  [**] [1:10000002:2] TCP packets detected, POSSIBLE SYN FLOOD [**] [Priority: 0] {TCP} 10.0.0.62:80 -> 10.0.0.25:2496
08/13-18:18:27.581127  [**] [1:10000002:2] TCP packets detected, POSSIBLE SYN FLOOD [**] [Priority: 0] {TCP} 10.0.0.25:2497 -> 10.0.0.62:80
08/13-18:18:27.581127  [**] [1:10000002:2] TCP packets detected, POSSIBLE SYN FLOOD [**] [Priority: 0] {TCP} 10.0.0.25:2498 -> 10.0.0.62:80
08/13-18:18:27.581128  [**] [1:10000002:2] TCP packets detected, POSSIBLE SYN FLOOD [**] [Priority: 0] {TCP} 10.0.0.25:2499 -> 10.0.0.62:80
08/13-18:18:27.581128  [**] [1:10000002:2] TCP packets detected, POSSIBLE SYN FLOOD [**] [Priority: 0] {TCP} 10.0.0.25:2500 -> 10.0.0.62:80
```

ERRORS AND CHALLANGES:

- CHALLANGE 1 (installation using Microstack)
    - During the installation of openstack using microstack we faced an issue while creating instance of cirrOS.
        - Error 1: ***Build of image xyz-abc... aborted, invalid input received unable to access the image after 3 tries.*** This error was solved using the link [3]
        - Error 2: ***Build of image xyz-abc... aborted, Volume xyz-abc.. Cannot be created after 3 tries.*** We couldn't solve this error, so we moved to Devstack as it is more stable.
- CHALLANGE 2 (Using Devstack)
    - ***openstack instances and services used lots of memory resulting into BareMetal (laptop) freezing, and upon restarting the device, few of openstack services failed to start such as network interface or cinder service due to which dashboard was not able to fetch instance details.***
    - ***To resolve this issue, we had to restart all the services and in worst cases had to reinstall openstack. [4]***

REFERENCES:

1. https://www.linuxfordevices.com/tutorials/ubuntu/install-openstack-on-ubuntu
2. https://www.kali.org/tools/hping3/
3. https://discourse.charmhub.io/t/microstack-invalid-image-identifier-or-unable-to-access-requested-image-http-400/5643/2
4. https://stackoverflow.com/questions/61029848/jsondecodeerror-in-openstack-ubuntu-18-4
5. https://stackoverflow.com/questions/73338845/how-to-install-openstack-on-ubuntu-in-2022
6. https://web.archive.org/web/20220204115608/http://www.hping.org/
7. https://opendev.org/openstack/devstack
8. https://docs.openstack.org/nova/queens/index.html
9. https://nmap.org/book/scan-methods-connect-scan.html
10. https://www.snort.org/resources#documents
11. https://wiki.openstack.org/wiki/?_ga=2.16861433.1929308491.1692149892-473278101.1691941403