Concordia Institute for Information System Engineering (CIISE)

INSE 6620 – Cloud Computing Security and Privacy

Project Proposal:

Implementing Network Based Attack on OpenStack Cloud Platform

Submitted to:

Professor Lingyu Wang

Submitted by:

Team Members	Student ID
Md. Saiduzzaman	40256249
Bhavya Patel	40254222
Jenil Pansuriya	40256184
Md. Aminul Islam	40203451
Nikunj Pathak	40203713
Tejas Surani	40248859
Devina Shah	40238009
Mansoureh Navidpanahtoupkanlou	40221901
Jenish Patel	40256632

Table of Content

Roles and Contributions

SN	Role	Student ID	Name
1.	Building Platform for OPENSTACK	40256249	Md. saiduzzaman
2.	Preparing the Attack Environment	40254222	Bhavya Patel
3.		40256184	Jenil Pansuriya
4.	Trigger Network Based Attack	40203451	Md. Aminul Islam
5.		40203713	Nikunj Pathak
6.		40248859	Tejas Surani
7.	Detect the Attack using SNORT	40238009	Devina Shah
8.		40221901	Mansoureh Navidpanahtoupkanlou
9.		40256632	Jenish Patel

Introduction

Modern computing is about protecting the complex digital ecosystems we navigate as well as executing daily activities with efficiency. The primary aim of the project revolves around the implementation of the cloud platform "OpenStack" and subsequently executing a network-based attack within this virtual environment. We explore OpenStack to understand why it's important and how it works. We also investigate the basics of Snort IDS (a system for detecting intrusions) to understand how it works too.

What is OpenStack?

A platform called OpenStack utilises shared virtual machines to create and administer private as well as public clouds. The "projects" or tools which make up the OpenStack architecture manage the core cloud computing functions of computations, storage, networking, identity, and image. In virtualization, a hypervisor divides services like archive, CPU, and RAM and abstracts them from multiple vendor-specific applications before delivering them as needed. [1]

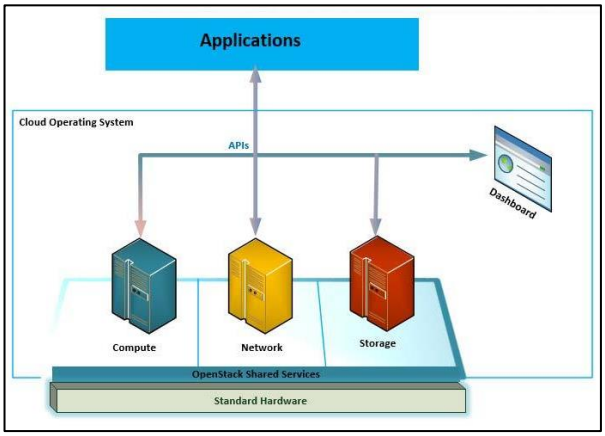

Fig 1.0 Standard Architecture of OpenStack

What is Snort?

Snort is widely recognized and open-source Intrusion Detection System (IDS) software that plays a pivotal role in monitoring network traffic and identifying potential security breaches, anomalies, or malicious activities within computer networks. Snort examines data packets traveling through a network and uses a combination of signature-based detection, protocol analysis, and anomaly detection techniques to raise alerts and offer details about potential threats, enhancing the overall security and robustness of network infrastructures. [2]

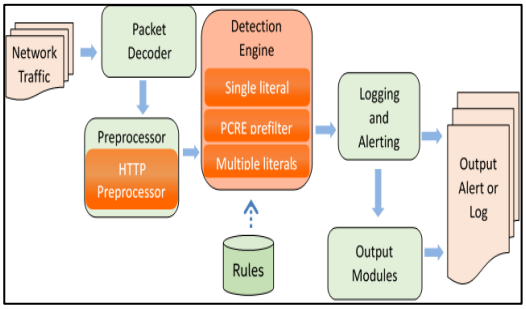

Fig 1.1 Standard Architecture of Snort

Network Based Attack and Detection using Snort

We conducted a DoS (Denial of Service) attack, which aimed to compromise the security of a targeted instance. To conduct this attack, we used the hping3 tool, known for its proficiency in generating such attacks. Our aim is to detect any kind of intrusion and possible network-based attack. By integrating the Snort Intrusion Detection System (IDS), we were able to actively identify and monitor the incoming attack traffic. Snort proved its effectiveness by detecting each ICMP ping echo request and echo reply.

Technological Stack

To perform the project we have used a physical host machine (Ubuntu 22.04 LTS) and on top of the bare metal machine, we installed OpenStack. Technical Stack details has been given below -

Host Node (Physical)

- Ubuntu 22.04 LTS (On DUAL BOOT)
- RAM – 10GB
- CPUs – 8 Core
- Disk – 50 GB

Target Node

- Platform – ubuntu 22.04 LTS
- Flavor – ds2G
- RAM – 2GB
- VCPUs – 2
- Disk – 20 GB

Hacker Node

- Platform – Kali Linux (Debian 2023.2
- Flavor – m1.small
- RAM – 2GB
- VCPUs – 1
- Disk – 20 GB

Tools

- Performing the attack - hping3 (version 3.0.0-alpha)
- Detection the attack - SNORT (version 2.9.15.1 GRE)

Fig 1.3 Details of Technical Stack

Operational Overview

These are the steps to install OpenStack with credential for Admin, Database, Rabbit and Services of OpenStack. Below image shows our OpenStack dashboard which shows Number of Instances, Allocation of VCPUs, RAM, Volume allocation, and Network utilities such as Security groups, Security rules, Ports, Routers, etc.[3]

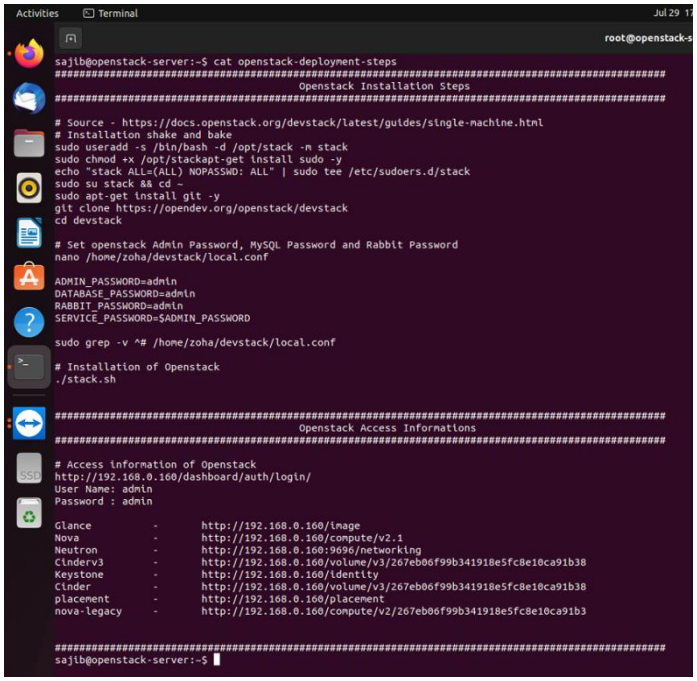

Fig 1.4 Operational execution of the platform with Network Topology diagram of OpenStack

This image shows the network topology with router making connection of public network with shared network (Internal Network) and such mapping helped us to access the instances outside the network. We can see details of these public networks and shared network in below image. We can see any interface that we add are shown here and it acts like a virtual router and do the routing between our nodes. We can create the security groups in Network topology section and apply any security policy that we need to. We can see the interface setting and set/edit the gateway IP.

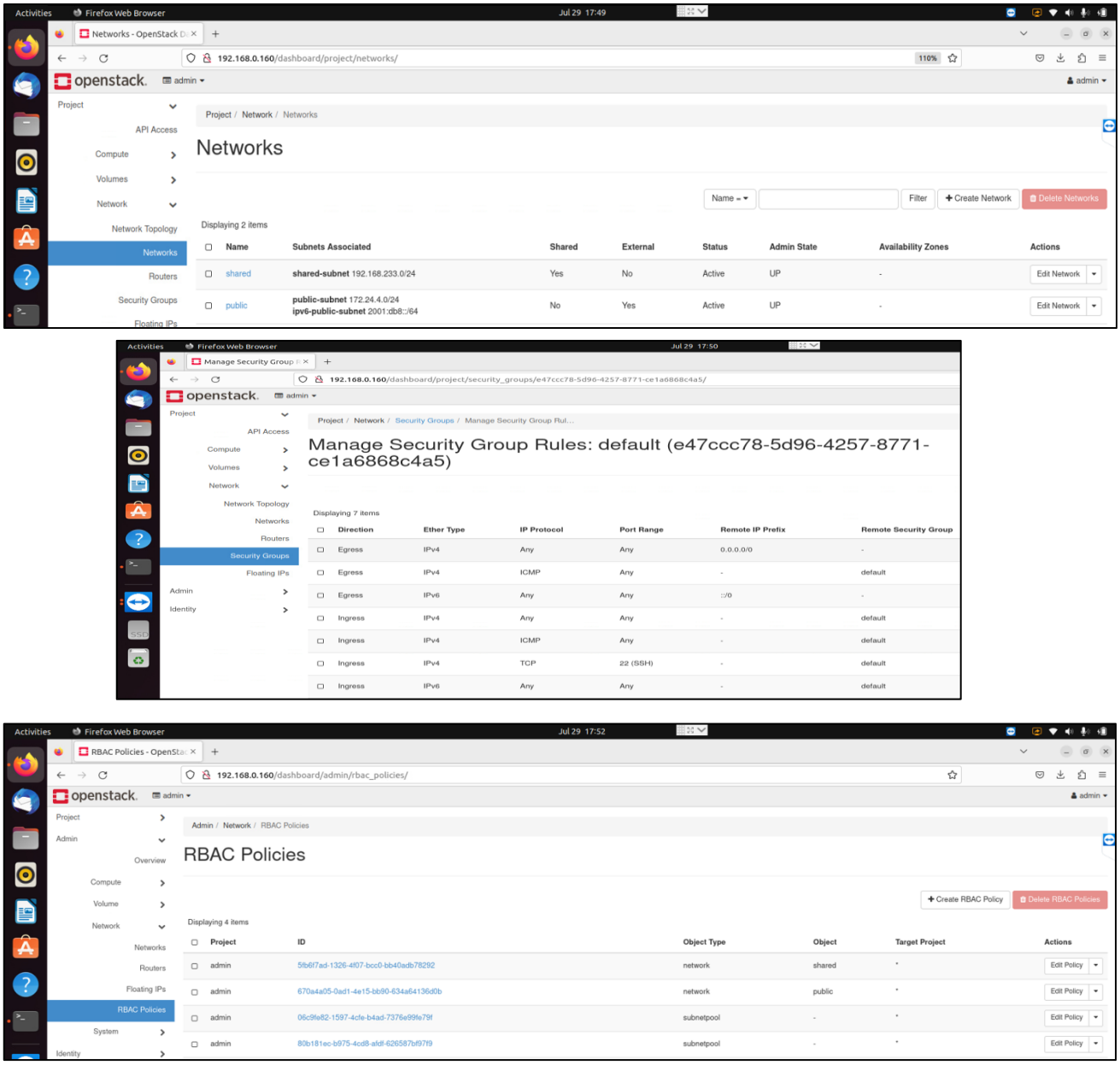

Fig 1.5 Network topology of routing on public network and shared network with Role Bases Access Control (RBAC)

Below image refers to our Hypervisor Summary. It shows details like Hostname and how much virtual resources have been utilised out of total allocated resource. For example, this image shows that we have utilised 4(1 for Application_node_1, 2 for **Target_node_01** and 1 for **Application_node_02**) out of total 8 VCPUs and we used almost half of the total allocated RAM, etc.

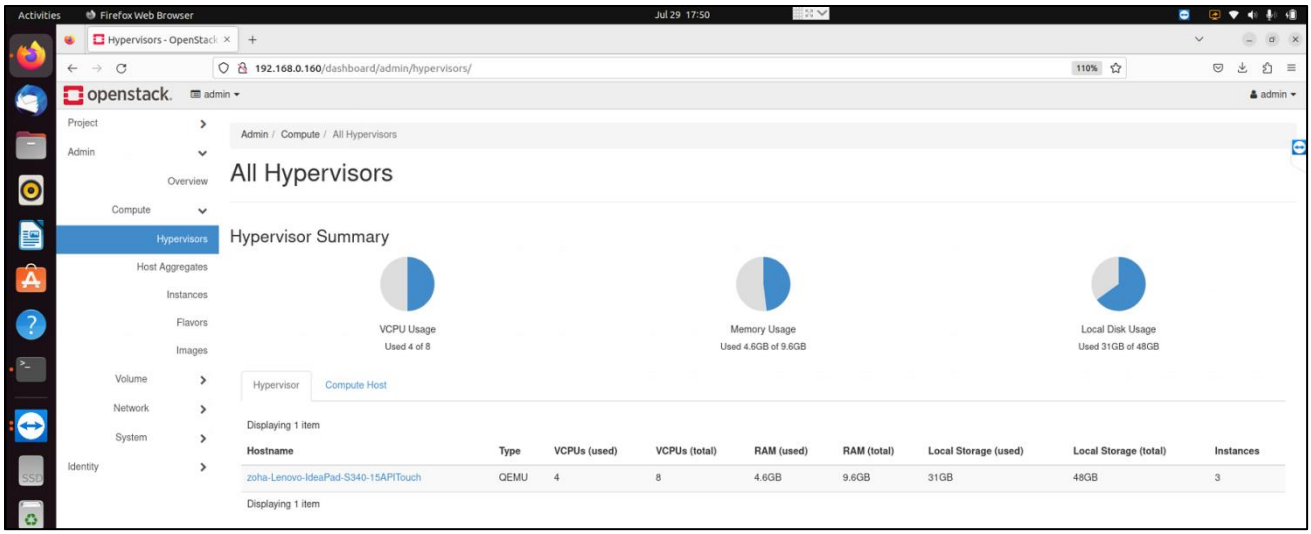

Fig 1.6 Details of Resource Allocation

We have three OpenStack images as per shown in below Image. In OpenStack, an "image" refers to a pre-configured and often virtualized operating system environment that serves as a template for creating virtual machines (VMs) or instances. These images encapsulate the necessary components, including the operating system, software, configurations, and even data, into a single file. When we want to create a new virtual machine in an OpenStack environment, we can choose an image that matches our desired specifications, such as the operating system type and version. By default, we have one image of CirrOS which is used for testing purposes. It is lightweight and efficient Linux distribution, so it is ideal choice due to its small size and rapid boot time.

OpenStack Component Details

Below image shows all the type we can select from for our instances. We can select any flavour according to different system requirements such as VCPUs, RAM, Root Disk, etc.

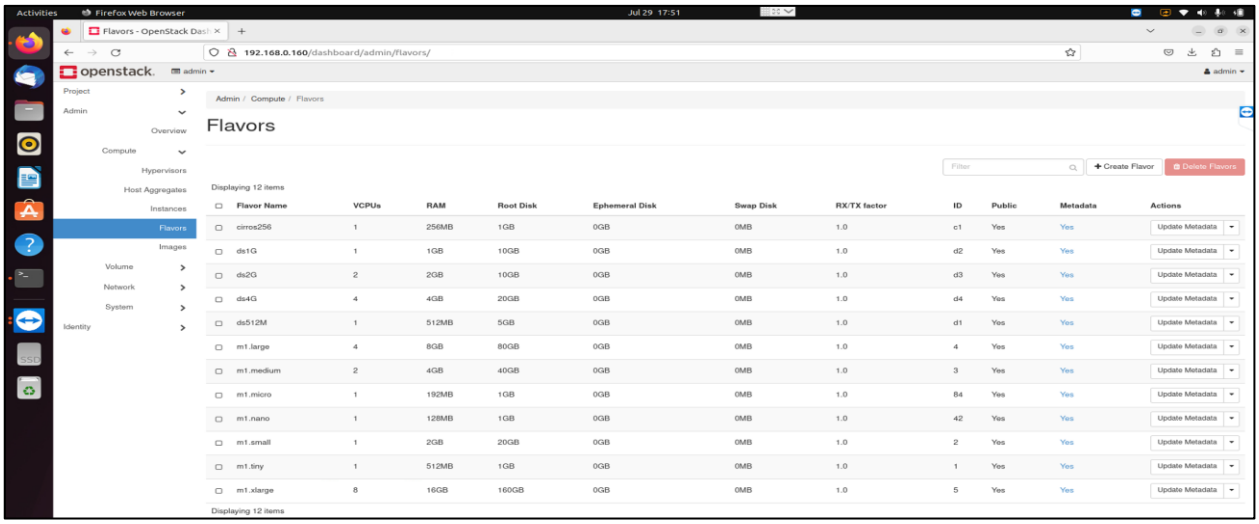

Fig 1.7 Instance Types with Resource template

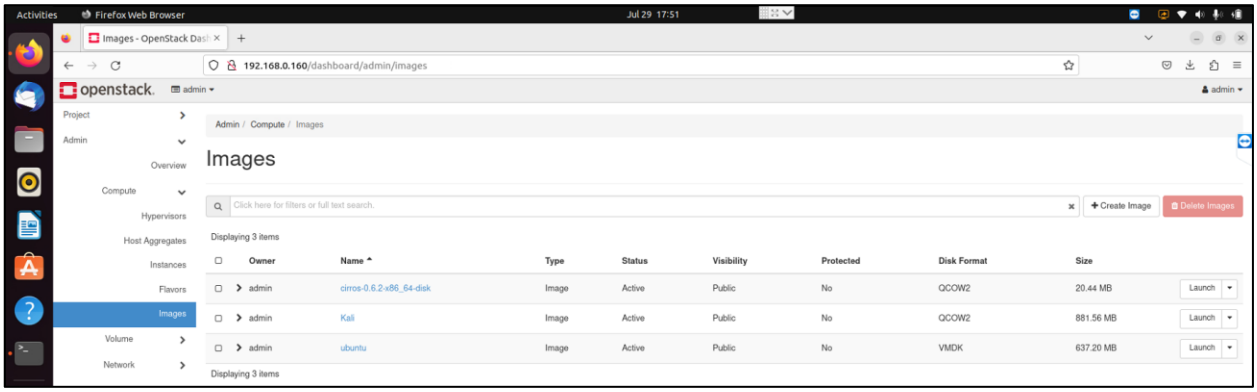

Fig 1.8 Image details with disk formats

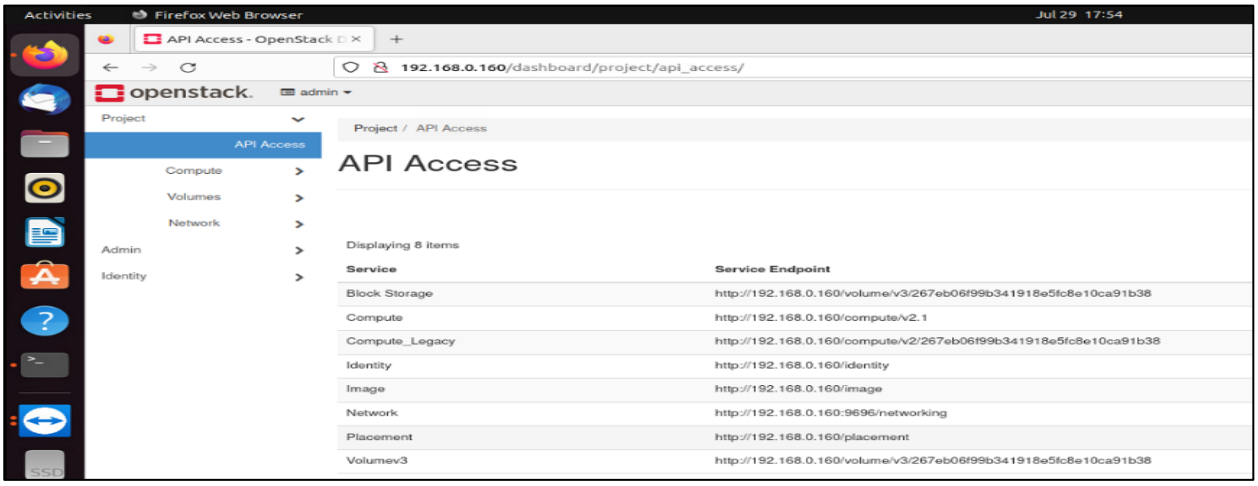

Fig 1.9 Service/Module wise API list

- These are the APIs that we can use as per our need that are provided by OpenStack.
- We can use these APIs to access and manage various services such as Block storage, Compute modules, identity module (keystone), Network, and volumes and other resources from OpenStack environment.
- Such APIs provide flexibility to users, administrators, and applications to programmatically create, modify and manage OpenStack cloud resources on demand.

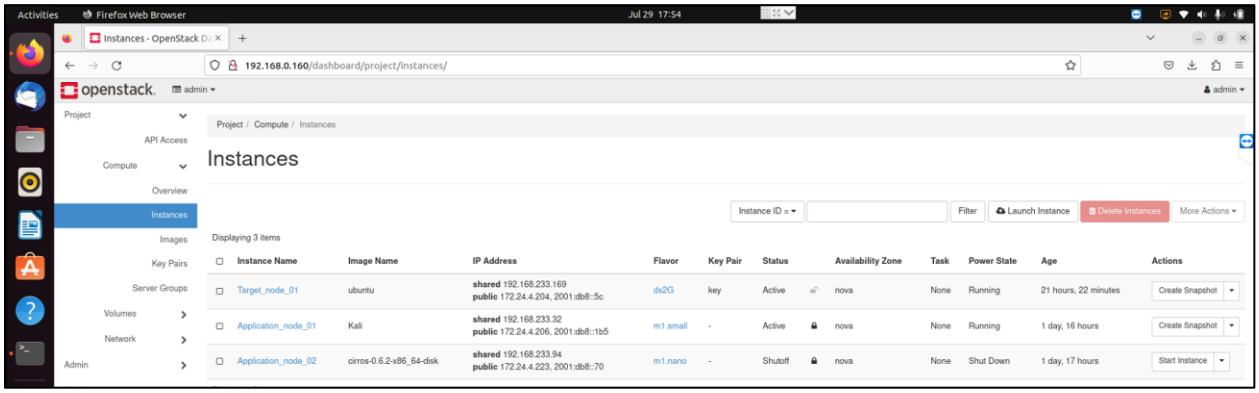

Fig 1.10 Instances List (Target node, Attackers Node & Test Node)

Methodology

In our project, we're working with two instances: Application_node_1, which serves as the attacker, and Target_node_01, our designated victim. To carry out the attacks, we've set up hping3 on **Application_node_01**. On the defensive side, we've equipped **Target_node_01** with Snort, a specialized tool to detect and respond to potential threats. We can see resources information of both the instances in following images –

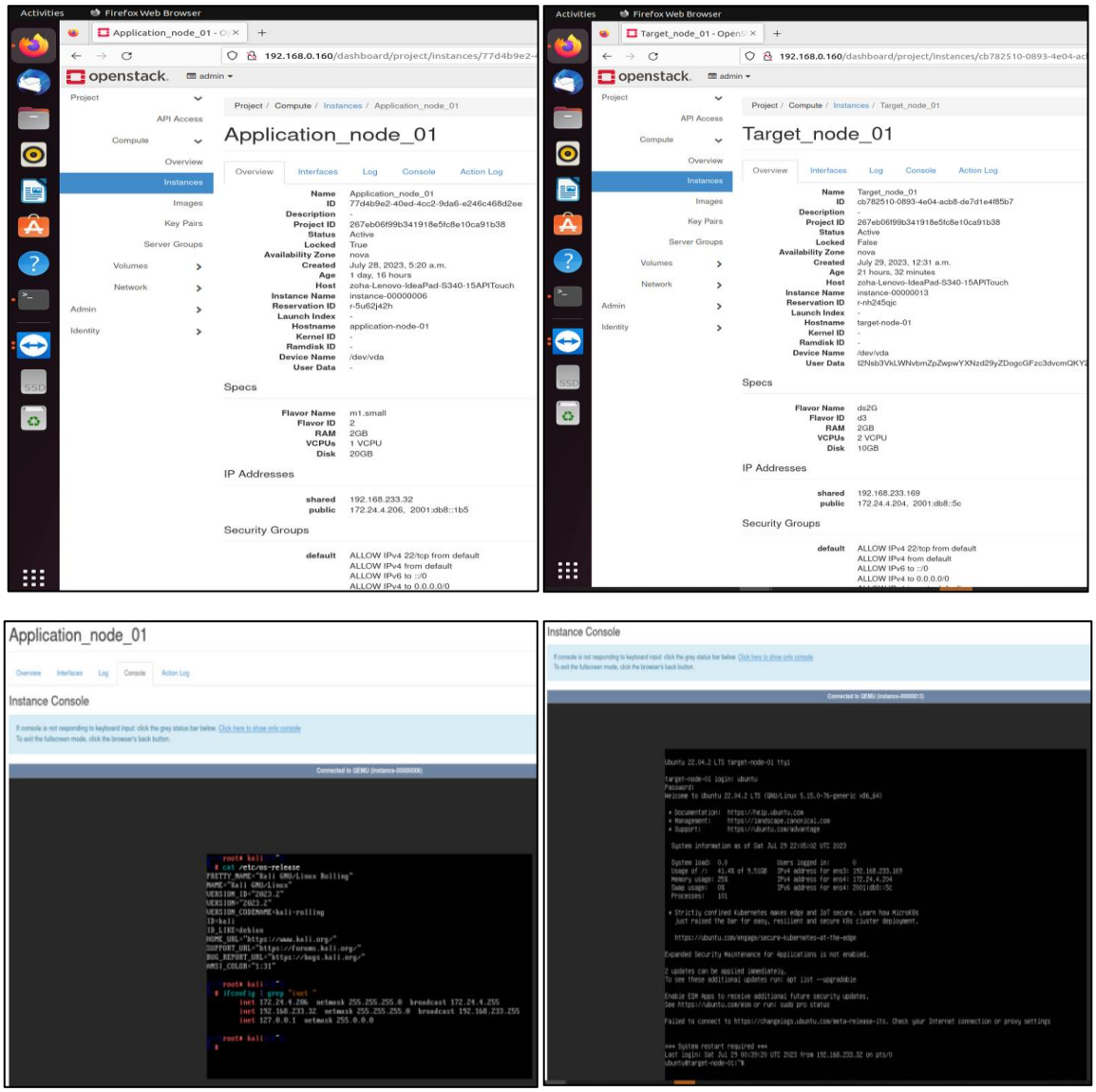

Fig 1.11 Instances resource details (Target node & Attackers Node)

Attack and Detection

In the following step, we sent single ping request to ensure that out IDS is correctly detecting network activity of our network cluster.

On the right side, we executed the hping3 command for the single ping request, while on the left side, the Snort defense tool showcased its capability. It successfully identified that the target node, the victim in our network, received an ICMP ping echo request dispatched by the attacker. Furthermore, Snort promptly responded by transmitting an ICMP ping echo reply to the source.

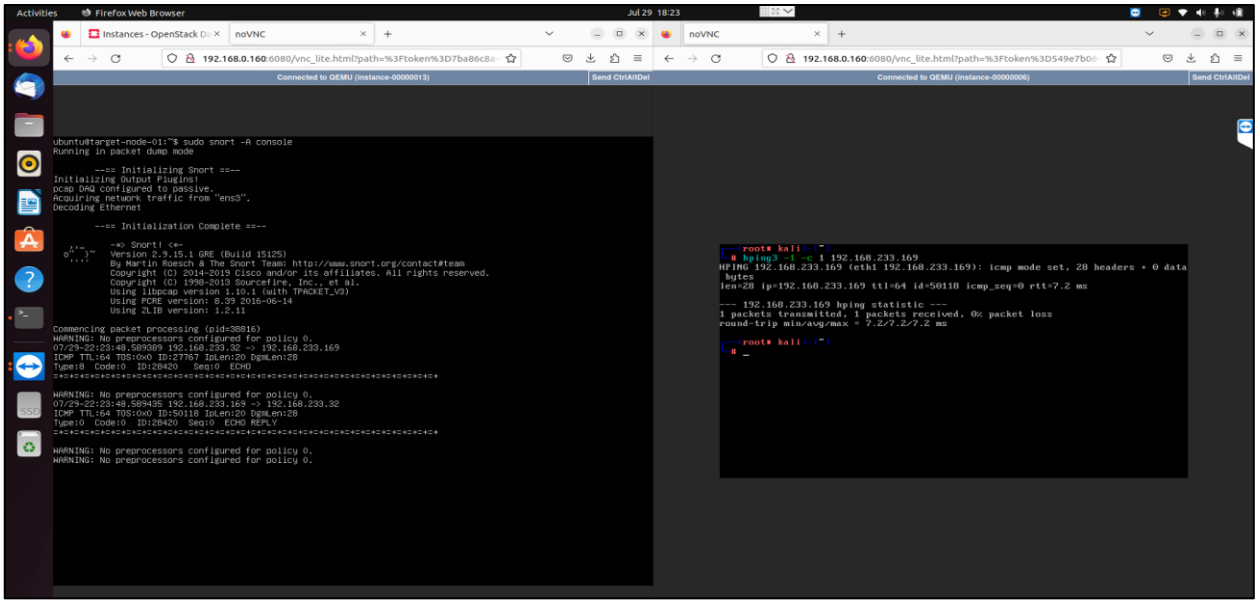

Fig 1.12 Single ping request to target using hping3 tool and detection using snort (IDS)

Types of attacks:

We have tried our attack with hping3 with three different options **--fast**, **--faster** and **--flood**. Each option dictates a specific rate of packet transmission:

--fast: it will send 10 packets per second.
--faster: it will send 100 packets per second.
--flood: it will send packets as fast as possible.

Here in the following three images, we are sending multiple ICMP ping requests to the victim at different speed rate and Snort is successfully detecting those requests and replying to that ICMP ping requests back to the sources.

Using the **"--fast"** and **"--faster"** flags can intensify the traffic load within the network, leading to a noticeable rise in the time taken for client responses. Conversely, opting for the **"--flood"** flag initiates an ongoing stream of packets, persistently bombarding the system until it becomes overwhelmed, resulting in an inability to effectively manage incoming requests. This can significantly impact the system's overall performance.

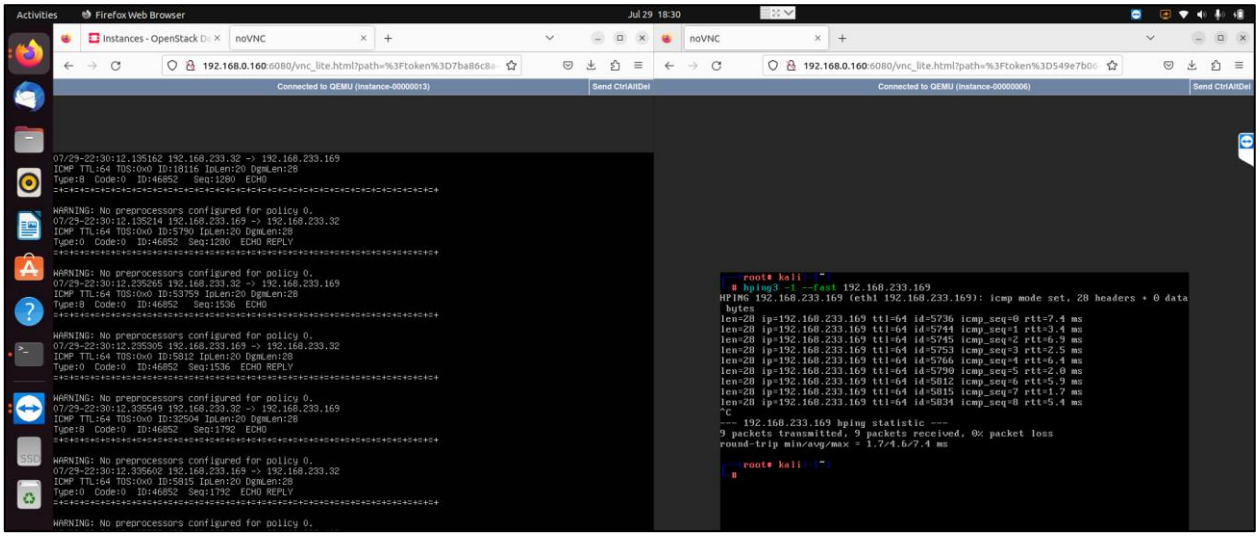

Fig 1.13 Multiple ping requests to target using hping3 tool and detection using snort (IDS)

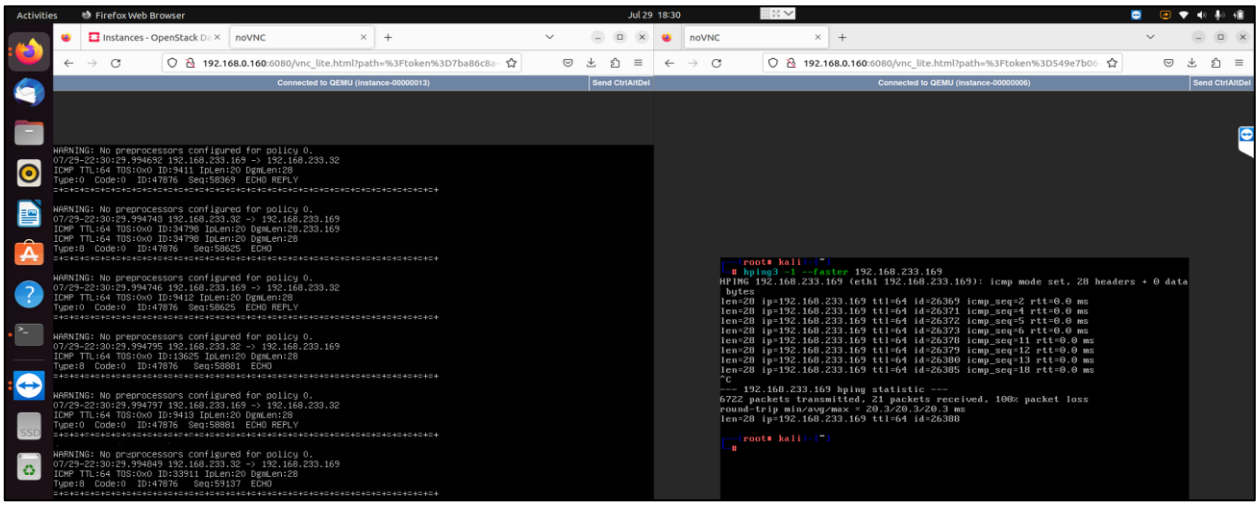

Fig 1.14 Detecting hping3 request with “FASTER” switch

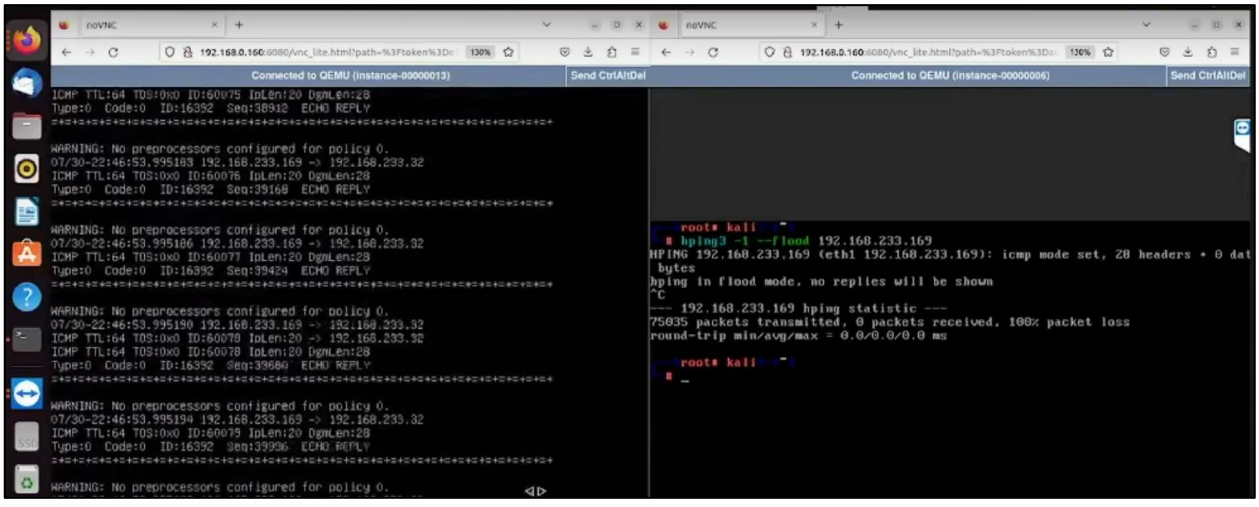

Fig 1.15 Detecting hping3 request with “FLOOD” switch

Challenges & Overcomes:

- Initially, we tried to install OpenStack on virtual machine but due to Hypervisor Type-2 complexity, we are unable to established network mapping with shared network [internal network of OpenStack/ local IP]. Hypervisor Type-2 Complexities we had faced during the installation process are -
    - Provision instances (OpenStack VMs) on top of Virtual Platform
    - Mapping network blocks
    - Routing issues
    - Processing delay

- To avoid network complexity in hypervisor type-2, we did perform dual boot on our laptop with 10GB RAM, 50GB SSD, corei7 9th generation on Ubuntu 22.04 LTS. By doing such dual boot, we avoid Hypervisor dependency; we successfully install OpenStack and implemented one attack on it and at the same time the original workstation of our classmate (which was on Windows OS remained intact).

- Due to scarcity of resources, we used single node of architecture instead of multi node architecture and we install IDS on the same instance of victim machine rather than making third instance to install IDS.

Reference:

- Available: https://www.redhat.com/en [Accessed: August-2023]
- Available: https://snort.org/ [Accessed: August-2023]
- Available: https://www.openstack.org/ [Accessed: August-2023]