Network Based Attacks and their Detection on Open-Stack Cloud Platform

Parth Vakil
Student ID: 40233132

Dennis Tank
Student ID: 40261560

Fahima Noor
Student ID: 40268465

Malvik Chauhan
Student ID: 40268733

Virenkumar Virani
Student ID: 40279337

Prashant Parmar
Student ID: 40291246

Peter Vourantonis
Student ID: 40157751

Abstract

This project aims to familiarize us with real-world cloud platforms while providing hands-on experience and knowledge dissemination. We used OpenStack as our cloud computing platform to build a small virtual network. This network includes establishing several nodes, with hosts specially designed for vulnerability attacks. We added Alpine OS instances to further our understanding and testing of lightweight environments in the OpenStack ecosystem. In addition, we used Snort as a separate node to monitor and detect potential network attacks. MicroStack assisted the deployment of OpenStack, making the setup process more streamlined and efficient. Throughout this project, we encountered issues with network configuration, security risks, and resource management. This study describes our setup, attack techniques, and detection procedures, as well as how we overcome these hurdles to achieve a safe and efficient virtual network environment.

***Keywords*—** OpenStack, Virtualization, Snort, Hping3, Dos Attack, Syn Flood, UDP FLood, Cloud, Network

Introduction

OpenStack is a cloud-based operating system designed to handle large amounts of computing, storage, and networking resources in a data center [1] [2]. It manages and allocates these resources using established authentication mechanisms via APIs. OpenStack includes a user-friendly dashboard that allows managers to monitor resources and users to customize them via a web interface. Aside from its fundamental infrastructure-as-a-service characteristics, it offers extra components for orchestration, problem tracking, and service management, guaranteeing that user applications are highly available. OpenStack, which is supported by industry titans including Rackspace, Dell, IBM, and Cisco, symbolizes the future of cloud computing [3].

Snort is a versatile open-source intrusion detection and prevention system that protects networks from a wide range of threats [4]. Snort functions as an intrusion detection system, continuously monitoring network traffic for anomalous patterns indicative of potential vulnerabilities or hostile incursions. It employs a rule-based framework to identify unusual activity, such as viruses and malware or efforts to gain unauthorized access [5]. In addition to being an identifying tool, Snort can be set up to actively block malicious traffic, acting as a preventative measure. Because of its versatility and real-time analysis, it is a fundamental component of modern network security systems, providing a strong defense against constantly changing cyber threats.

1 Environment Setup

1.1 Environment Details

1.1.1 System Specification:

CPU: 13th Gen Intel(R) Core(TM) i9-13900HX 2.20 Ghz
System Type: x64 based processor
RAM: 16.0 GB
Storage: 1TB M.2 PCIe NVMe SSD

1.1.2 Environmental Methodologies:

We used various ways to set up the environment, however, we faced many problems - which are covered in Section 3 - while executing the post-tasks in the process; further information is as follows:

- **MicroStack:** We proposed to use Microstack as a medium to install OpenStack and exercise the project [6].
- **DevStack:** The proposed method was not successful so we moved to DevStack for the installation of OpenStack [7].
- **Manual Installation:** In parallel to the DevStack and MicroStack we divided the tasks among the team to install the OpenStack from scratch [8].

The focused task was to execute attacks and detect them in the instances, as there were chances for the failure of one or more set-up methods we planned to use more than one approach.
The finalized approach was to use the OpenStack installed from the manual installation as we had more control over handling the errors/problems that were occurring in the MicroStack and DevStack.

1.2 OpenStack Installation

1.2.1 General:

The main idea is to use VirtualBox to install the nodes i.e. controller, compute and block.

- **Host OS:** Ubuntu Server 16.04 LTS
- **Network:**
  - Management Network (Host-only network): CIDR: 10.0.0.0/24; Gateway:10.0.0.1
  - Provider Network (NAT): CIDR: 203.0.113.0/24; Gateway:203.0.113.1
  - NetNetwork (NAT):
- **Host addresses:** Controller:10.0.0.11; Compute:10.0.0.31; Block:10.0.0.41

The management network is for connecting and syncing the nodes together, while the provider network is to connect the overall OpenStack with the internet which is important for the instances as they need web access to download and install required tools.

1.2.2 Controller Node:

VCPU (cores): 2
RAM: 6GB
Disk: 21GB
Network: VirtualBox Host-Only Network Ethernet Adapter
IPv4: 10.0.0.11
Services:

- **Network Time Protocol (NTP):** The Network time is an application layer protocol in the TCP/IP [refhuwai] that is used to synchronize the clocks in a cloud server and provide high-precision time correction. This allows us to ensure that the chef-client is working. It is needed as consistent timestamps make log analysis across different nodes simpler [9].

- **MYSQL Database:** Many Openstack services need to store information in a database. Thus a database is needed. We ran the MySQL database in our controller node [10].
- **RabbitMQ:** OpenStack uses message queuing to enable different components to communicate [refrabbit1]. RabbitMQ provides message queues for peer-to-peer communication. Message queues effectively facilitate command and control functions across OpenStack deployments [11].
- **Memchached:** Memcached is a high-speed memory-based storage system [12]. It boosts application performance by storing frequently accessed data like database results, API responses, or website content directly in the server's memory, reducing read load on the database.
- **ETCD:** ETCD is a distributed reliable key-value store for distributed key locking, storing configuration, keeping track of service live-ness, and other scenarios [13].

OpenStack components:

- **Keystone:** Keystone is an identification and authentication service that verifies the identity of the user and also provides information on the resources that can be accessed by the user [14].
- **Glance:** Glance is an image service that allows users to discover, retrieve, and register Virtual Machine (VM) images and container images [15].
- **Horizon:** Horizon is OpenStack's primary web interface. Initially focused on managing compute resources, it has expanded to cover multiple OpenStack services. Horizon's development emphasizes core functionality, flexibility, manageability, consistency, stability, and user-friendliness [16].

Horizon Interface: 10.0.0.11/Horizon

1.2.3 Compute Node:

VCPU (cores): 1
RAM: 7.8GB
Disk: 44GB
Network: VirtualBox Host-Only Network Ethernet Adapter
IPv4: 10.0.0.31
OpenStack components:

- **Nova:** Nova is the OpenStack component responsible for creating and managing virtual machines. It provides a platform for quickly provisioning and controlling the virtual servers [17].
- **Neutron:** Neutron is a networking service that manages the network connections between virtual machines (created by Nova) and other cloud resources [18].

1.2.4 Block Node:

VCPU (cores): 1
RAM: 2GB
Disk1: 20.13GB
Disk2: 36.77GB
Network: VirtualBox Host-Only Network Ethernet Adapter
IPv4: 10.0.0.41
OpenStack components:

- **Cinder:** Cinder is a block storage service that provides volumes to Nova virtual machines, bare metal hosts, and containers [19].

1.3 Setting Up Virtual Network

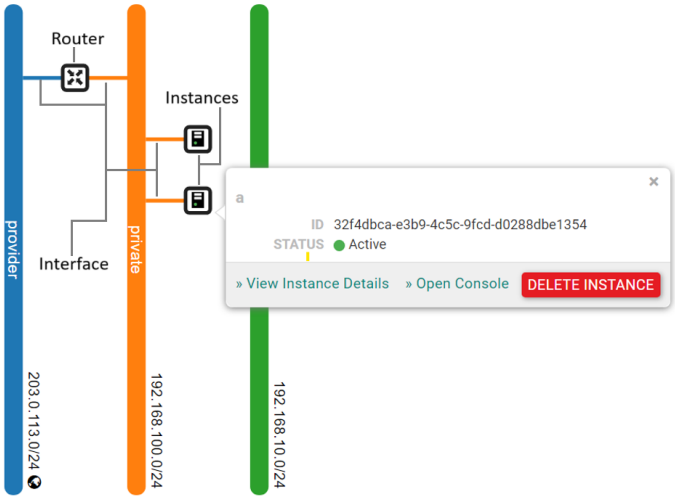

Fig. 1: Network Topology

Observation:
The diagram Fig. 1, illustrates a simplified OpenStack network configuration. The blue link represents the provider network, which has direct internet connectivity. Conversely, the orange link signifies the private network, isolated from the public internet. A virtual router acts as a bridge between these two networks, facilitating communication between them. The interfaces, depicted as connections between network components, enable the transmission of data within the network.

Provider Network:

- **Description:** This network line is connected to the outside internet.
- **Adaptor:** VirtualBox NAT
- **CIDR:** 203.0.133.0/24

Private Network:

- **Description:** We created a private network with sub-netting enabled. This is the main virtual network line where all the instances will be connected via interface links.
- **CIDR:** 192.168.100.0/24

Router:

- **Description:** It is a virtual router that will act like a real-world router as a gateway to the worldwide internet. It will be connected with two interfaces.
- **Access-Point:** 192.168.100.1
- **Broadcast:** 192.168.100.255

Interfaces:

1. $ProviderNetwork \leftrightarrow Router$
2. $Router \leftrightarrow PrivateNetwork$
3. $PrivateNetwork \leftrightarrow Instance1(detector)$
4. $PrivateNetwork \leftrightarrow Instance2(attacker)$

1.4 Instance Creation

Attacker

- **OS;** Ubuntu 16.04 Cloud Image configured with default credentials as root.
- **VCPU:** 2
- **RAM:** 512MB
- **DISK:** 5GB
- **IPv4:** 192.168.100.6

Detector

- **OS;** Ubuntu 16.04 Cloud Image configured with default credentials as root.
- **VCPU:** 6
- **RAM:** 2GB
- **DISK:** 10GB
- **IPv4:** 192.168.100.9

2 Attack and Defence

2.1 Overview

For the OS we primarily selected the Alpine Server image for the Instances, however, the OS is stateless i.e. the os state resets on every restart. Additionally, we faced difficulty setting up many Linux packages and tools, including snort; many packages aren't available on the standard 'repo' link. Further, we were expecting the snort to come with community rules in the package, but the snort 'apk' in Alpine is lightweight and doesn't come with community rules, which created warnings and errors in the '.confi' file while running the snort. We Finalized the Ubuntu image and installed Snort and all the required tools for the attack and detection.

At the Attacker side we used tools as follows:

- Python3 Scapy Scripts
- Go Executives
- Nmap
- Hping3

At the Defender Side we used tools as follows:

- Python3 - for HTTP server
- Snort
- Tshark

2.2 SYN Flood - DOS

Attack command: *hping3 —-rand-source 192.168.100.9 -p 80 -S -–flood*

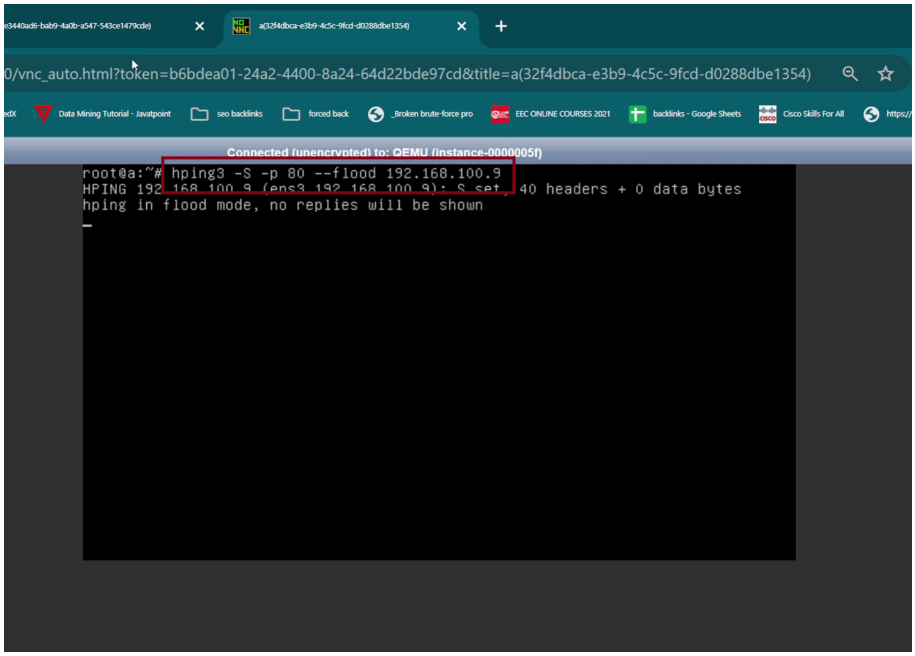

Fig. 2: SYN Flood Attack

Snort Rule: *alert tcp any any → $HOME_NET 80 (msg:"SYN FLOOD"; detection_filter: track by_dst, count 20, seconds 60; sid:10000009; rev:2;)*

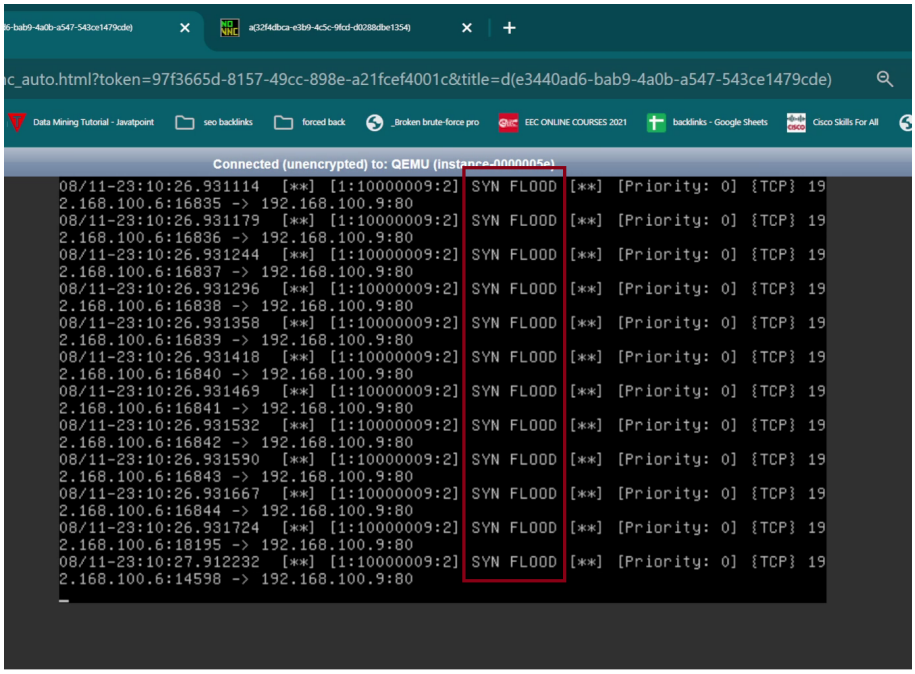

Fig. 3: SYN Flood Detected

2.3 UDP Flood - DOS

Snort Rule: *alert udp any any → $HOME_NET 53 (msg:"UDP FLOOD"; detection_filter:track by_src, count 10, seconds 60; sid:10000007; rev:3;)*

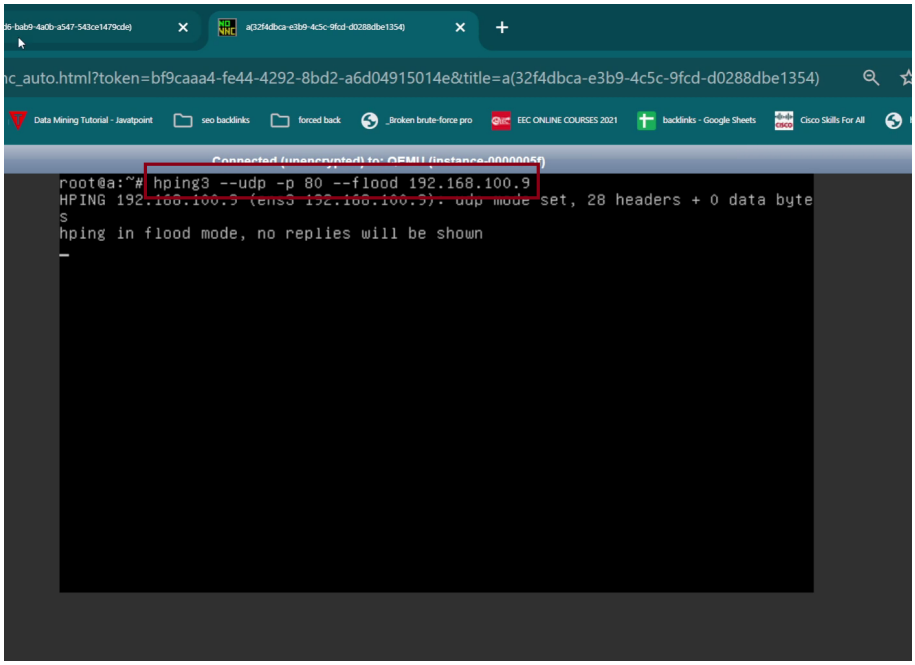

Fig. 4: UDP Flood Attack

Attack command: *hping3 -2 —-flood —-rand-source -p 53 192.168.100.9*

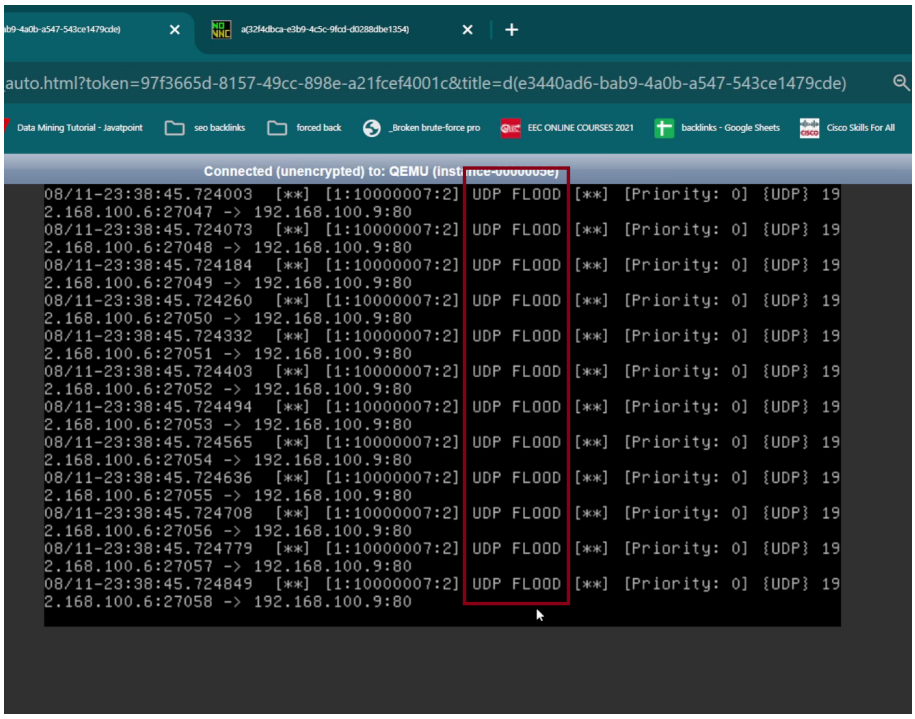

Fig. 5: UDP Flood Detected

2.4 UDP Scan Attack

Snort Rule: *alert udp $EXTERNAL_NET any → $HOME_NET 53 (msg:"UDP SCAN"; detection_filter:track by_dst, count 20, seconds 60; classtype: attempted-recon; sid:10000006; rev:2;)*

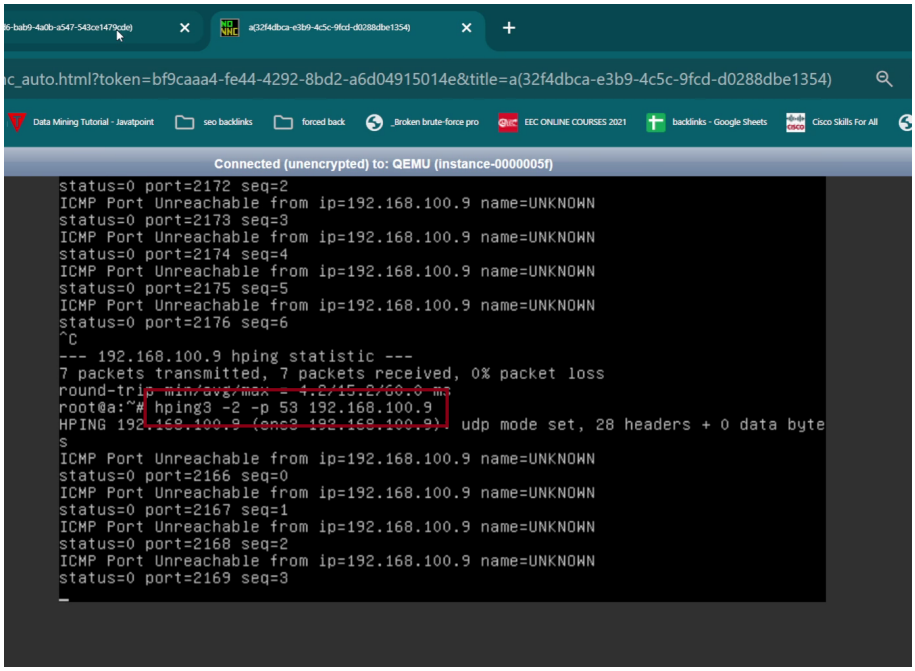

Fig. 6: UDP Scan Attack

Attack command: *hping3 -2 192.168.100.9 -p 53*

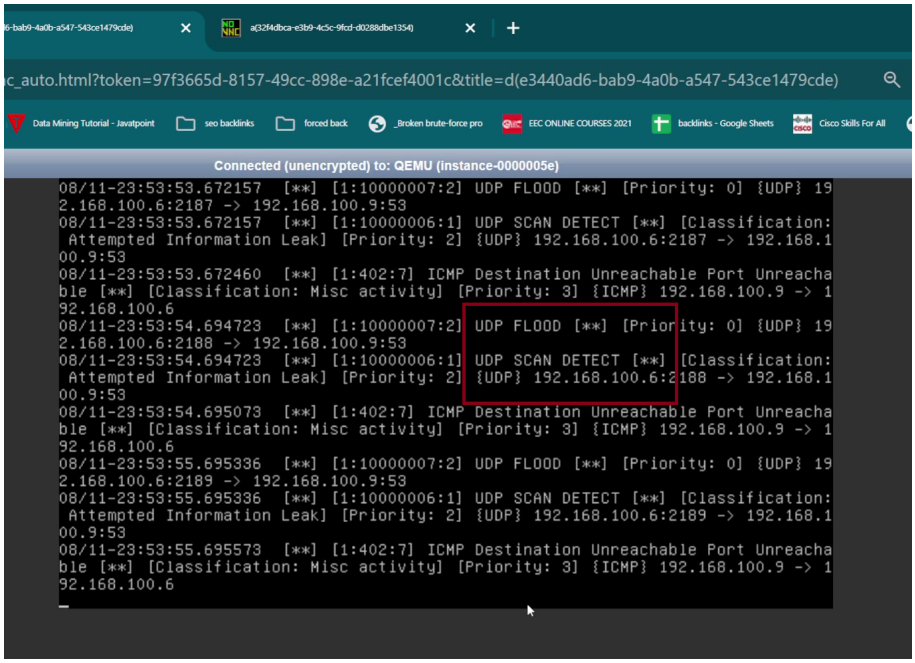

Fig. 7: UDP Scan Detected

3 Challenges

3.1 MicroStack

MicroStack is a compact solution to get OpenStack installed in the system and start the instances. We planned to install MicroStack in the VirtualBox and were able to successfully do so. However, the cinder backend service was not synced properly with the controller. The point of the note here is that, in MicroStack the nodes are working in containers and cinder is responsible for creating Volumes for the virtual instances. We were able to get the

cinder-scheduler.log file and we found that the issue was with the cinder API service. We tried re-configuring cinder and restarting the service [20], and we tried reinstalling the MicroStack and running it with the required flags [21], however, the error did not stop so we planned to move to DevStack installation.

3.2 DevStack

Our initial challenge with Devstack was a hardware virtualization compatibility issue, which we were able to resolve by turning off the core isolation. Next, we encountered network connectivity issues caused by an incorrectly configured NAT. We also had problems with the SQLAlchemy plugin which we fixed by updating the plugin to its latest version. Even after doing so, there were SQLAlchemy association proxy issues that required a local code modification [22]. Despite resolving numerous Devstack issues, we were unable to progress beyond the setup phase. Installing Devstack in VirtualBox and configuring a host-only network adapter failed to provide instances with internet connectivity [23].

3.3 Manual Installation

Given the persistent challenges encountered with Devstack, we decided to carry out manual OpenStack installation. Firstly, we created the Virtualbox using the latest Ubuntu server image. Unfortunately, a manual OpenStack installation proved equally problematic. The default image's behavior of selecting all available network interfaces hindered the installation process. Moreover, the absence of essential utilities like vi and net tools within the image presented significant challenges in managing network configurations as desired. Going through the OpenStack website for support, the links provided were for Ubuntu server 16.04 and thus made us shift to that version.

After installing Ubuntu, the vi was not working properly, so we had to make a /.vimrc file and put some settings in it to make the vi work properly. We also face issues deploying Keystone. There were problems with the hostname used and commands. The problem was resolved when we changed the hostname in /etc/hostname and /etc/hosts, it worked.

We also faced a problem while detecting compute node 1. When checking nova-compute logs, we found that it was having a problem while authenticating to rabbitmq. It was later found that user creation for rabbitmq on the controller node was not done successfully. After doing it the problem was solved.

After setting up Horizon, we got an invalid service catalog network error. This happened because we mistakenly created two network services when troubleshooting issues in Neutron. After deleting the second network service, the issue was resolved.

Lastly, we faced problems While creating flavors and deploying instances. The Flavors were larger than any single compute node. Because of that, the Nova scheduler was not able to schedule the instance on any one Compute node (Even if the instance con-fig is less). Due to this, A compute 3 Node was created with much better con-fig, and Compute 1 and 2 were powered off.

3.4 Instance

Initially, we planned to use the Alpine image for both instances and with this decision, we faced an issue of fewer community packages for the Alpine repository. When we were using the new version of Alpine images the instances were stuck on boot and were not loading further, while the older version of Alpine images were booting successfully. While the latest Alpine versions do have a good community the older versions do not we were unable to install snort or attacking tools in both instances.

3.5 Snort

Two primary challenges emerged while utilizing Snort. Firstly, the deprecated 'threshold' parameter [24] in existing community rules necessitated the creation of new rules employing current alternatives 'detection_filter' [25]. Secondly, Snort's inability to detect packets was resolved through the implementation of a generic traffic alert and subsequent packet confirmation using Tshark.

3.6 Attack

Attempts at attacking the detection nodes using Python and Go scripts were unsuccessful due to performance issues and system instability. The execution turned out to be very heavy and crashed the whole system. Subsequently, Hping3, Nmap, and Nping were employed, which resulted in successful attacks against the detection nodes.

Conclusion

The project aimed to establish an OpenStack environment to simulate a network-based attack and its detection using Snort. While the initial setup phases presented significant challenges with various OpenStack distributions, manual installation ultimately proved successful. The process of selecting suitable OS images for the virtual instances also involved experimentation before settling on Ubuntu's Xenial cloud image. Snort, equipped with community rules, was successfully installed on one instance to function as an intrusion detection system. However, the deprecated 'threshold' parameter in Snort rules initially hindered the detection of attacks, necessitating the use of the 'detection_filter' parameter. The project encountered obstacles in both the attack and detection phases, requiring multiple iterations with different attack tools and Snort rule adjustments. While the core functionalities of OpenStack, instance creation, and Snort installation were achieved, the intended goal of reliably detecting simulated attacks using Snort was partially realized due to the complexities involved in rule configuration and attack behavior. Further research and experimentation with Snort rule sets, attack methodologies, and network configurations are recommended to enhance the effectiveness of the attack detection system.

Acknowledgment

We would like to express our sincere gratitude to Professor Lingyu Wang for his invaluable guidance and support throughout this project. His expertise in the field of OpenStack was instrumental in shaping the direction of our research. His suggestion to manually install OpenStack, despite the potential challenges, proved to be pivotal in achieving our project goals. We are also grateful for his advice to utilize DevStack as a backup plan, which provided a valuable contingency option. We are indebted to his encouragement and constructive feedback, which significantly contributed to the successful completion of this project. Additionally, we would like to acknowledge the support and resources provided by Concordia University and its Institute of Information System Engineering for facilitating this research endeavor.

Contribution Table

Member/ Tasks	**Enviroment Setup**			**Instance Setup**		**Snort Rules**	**DOS Attack**
	Microstack	**DevStack**	**Manual Installation**	**OS testing and Selecting**	**Tools Installation**		
Parth Vakil		***YES***	***YES***	***YES***			
Dennis Tank		***YES***	***YES***				***YES***
Fahima Noor	***YES***	***YES***		***YES***			
Malvik Chauhan		***YES***	***YES***			***YES***	
Prashant Parmar	***YES***				***YES***		***YES***
Peter Vourantonis				***YES***	***YES***	***YES***	
Virenkumar Virani	***YES***				***YES***		***YES***

References

[1] Openstack. https://www.openstack.org/.

[2] Shweta Gumaste, DG Narayan, Sumedha Shinde, and Amit Kachavimath. Detection of ddos attacks in openstack-based private cloud using apache spark. *Journal of Telecommunications and Information Technology*, 4:62–71, 01 2021.

[3] Rackspace and dell emc team to revolutionize private cloud. https://www.rackspace.com/en-gb/newsroom/rackspace-dell-emc-team-revolutionize-private-cloud, 2017.

[4] Snort. https://www.snort.org/.

[5] Snort rules. https://www.sapphire.net/blogs-press-releases/snort-rules-examples/.

[6] Install openstack using ministack in ubuntu. https://gopalkumr.medium.com/install-openstack-using-ministack-in-ubuntu-23b4084f7452, 2024.

[7] Devstack. https://docs.openstack.org/devstack/latest, 2023.

[8] Openstack installation and deployment. https://concordia.udemy.com/course/openstack-installation-and-deployment/?utm_campaign=%5B%27email%27%5D&utm_medium=%5B%27email%27%5D&utm_source=%5B%27sendgrid.com%27%5D, 2018.

[9] Ntp service. https://documentation.suse.com/soc/9/html/suse-openstack-cloud-crowbar-all/cha-depl-nostack.html.

[10] Mysqldatabase. https://www.mysql.com/.

[11] Messaging queuing. https://docs.redhat.com/en/documentation/red_hat_openstack_platform/13/html/security_and_hardening_guide/message_queuing#message_queuing.

[12] Mysqldatabase. https://memcached.org/.

[13] Etcd ubuntu. https://docs.openstack.org/install-guide/environment-etcd-ubuntu.html, 2024.

[14] Keystone. https://docs.openstack.org/keystone/latest/, 2024.

[15] What is openstack glance? https://openmetal.io/docs/glossary/what-is-openstack-glance/, 2024.

[16] What is openstack horizon? https://openmetal.io/docs/glossary/what-is-openstack-horizon/, 2024.

[17] Nova. https://launchpad.net/nova, 2024.

[18] Neutron. https://wiki.openstack.org/wiki/Neutron, 2024.

[19] What is openstack cinder? https://openmetal.io/docs/glossary/what-is-openstack-cinder/, 2024.

[20] Cinder volumes fail to create when using a qcow2 image openstack. https://stackoverflow.com/questions/68938341/cinder-volumes-fail-to-create-when-using-a-qcow2-image-openstack-packstack, 2021.

[21] Microstack won't let me create volumes. https://www.reddit.com/r/openstack/comments/12xbyzo/microstack_wont_let_me_create_volumes_or/, 2023.

[22] Update sqlalchemy and alembic, drop sqlalchemy-migrate. https://review.opendev.org/c/openstack/requirements/+/879743?tab=comments, 2023.

[23] Any advise how to get access to internet from instances? https://github.com/lorin/devstack-vm/issues/2, 2013.

[24] Snort readme.thresholding. https://www.snort.org/faq/readme-thresholding, 2024.

[25] Snort readme.filters. https://www.snort.org/faq/readme-filters, 2024.