



INSE - 6620 Cloud Computing Security and Privacy

Final Project Report

Summer Term 2022

Course Instructor: Dr. Lingyu Wang

Project Topic:

OpenStack Cloud Deployment and Network Security Analysis

Team Members:

Himanshu Mahajan	40193970
Varnesh Gawde	40159406
Sushant Padmanabhi	40164604
Mehul patil	40191499
Harish Krishna Venkateswaran	40184965
Vasu gandhi	40170539
Tanmya Rampal	40197691

Table of Contents

1. Introduction	1
2. Literature review	2
3. Implementation	4
4. Attacker and Victim Instance	9
4.1. Snort Introduction	9
4.2. Snort Installation and Configuration	11
4.3. ICMP Attack	12
4.4. SSH Failed Login	13
7. Obstacles	13
8. Conclusion	14
9. Contributions	
References	

1. Introduction

Cloud computing boomed parallelly along the internet with the increase in the need for storage and computations. As cloud computing was the on-demand availability of computer system resources, primarily data storage (cloud storage) and computing power, without direct active supervision by the user, it played a crucial role in the emergence of new IT enterprises with smaller budgets and uncertain use cases. Companies save a lot of money when using cloud computing as it follows the pay-as-you-go model, which restricts overpaying of unused resources.

The definition goes by,

A model called "cloud computing" makes it possible to quickly create and deploy a shared pool of reconfigurable computing resources (such as networks, servers, storage, applications, and services) via a network while requiring little management labor or service provider interaction [1].

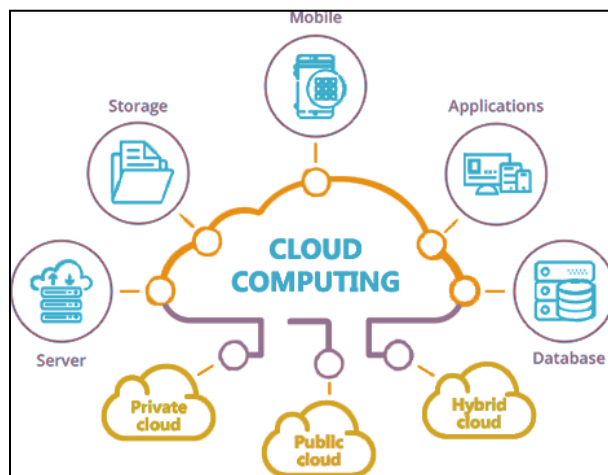


Fig 1. Cloud Computing [Source]

The five important properties of cloud computing are:

- On-demand self-service
- Broad network access
- Resource pooling
- Rapid elasticity
- Measured Service

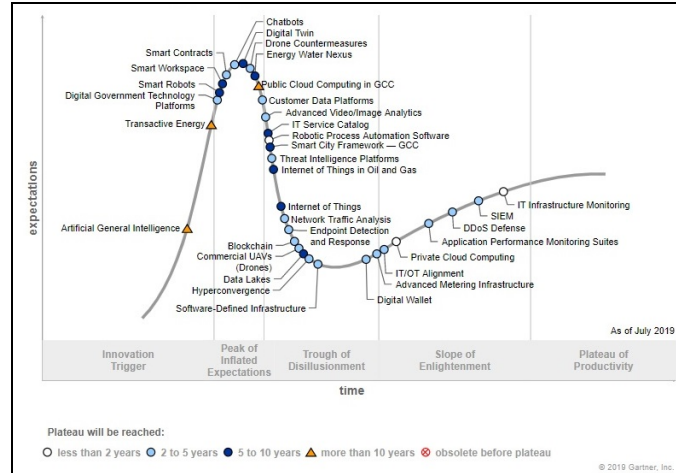


Fig 2. Popularity of cloud computing overtime

As cloud computing became more popular, more companies started using the technology, without much thinking about the security aspect of the technology which is considered a primary obstacle to the wide adoption of clouds. The security sector faces additional difficulties as a result of the novel ideas that clouds provide, such as resource sharing, outsourcing, and multi-tenancy. As in the cloud, functionalities are distributed across data centers, and a lot of computation of important data is conducted in places where much other information is stored as well. This leads to whether “the data is safe or not?”.

In order to assess the perceived level of security in the cloud, we attempted to implement cyberattacks from one cloud node to another node in this project and detect the associated attacks at the victim node. The tools used for the implementation purposes are:

- OpenStack
- Snort
- VMWare

Keywords: *Cloud computing, Information Technology, Data Centers, pay-as-you-go.*

2. Literature review

It is obvious that the biggest obstacle to cloud computing has been the security issue. Without a question, many people find the idea of storing their data and running their program on someone else's hard drive while using their CPU to be intimidating. The data and software of a business are seriously threatened by well-known security problems including data loss, phishing, and botnets (operating remotely on a network of computers). Additionally, the cloud computing multi-tenancy model and shared computing resources have created new security concerns [2] that need cutting-edge techniques to address. For instance, hackers intend to set up botnets using the cloud since it frequently offers more dependable infrastructure services at a relatively lower cost for them to launch an assault [2].[\[source\]](#)

It is always seen throughout history that whenever a new technology is deployed “black hats” have proved to break it and make it vulnerable to the open world. Cloud computing faced the same issue when it was introduced first, with the intention to save cost and make it flexible, the technology has a lot of loopholes in terms of security. The properties that made cloud computing popular were the reasons for its security flaws. Talking about the project, we have made use of OpenStack. Open source cloud computing has attracted a large number of user groups by the advantages of open source and low cost and has now become a large-scale promotion and application.[<https://iopscience.iop.org/article/10.1088/1755-1315/69/1/012140>]

NASA and Rackspace jointly developed OpenStack, a free piece of software with an Apache-licensed open source project. The goal of OpenStack is to offer service support for almost all varieties of cloud systems. This project's objective is to provide a cloud computing

A management platform that is easy to scale, practical to use, and effective to standardize. Users of OpenStack are provided with fundamental services via ancillary services, each of which can be integrated into an API foundation [3]. Figure{3} depicts the architecture of OpenStack.

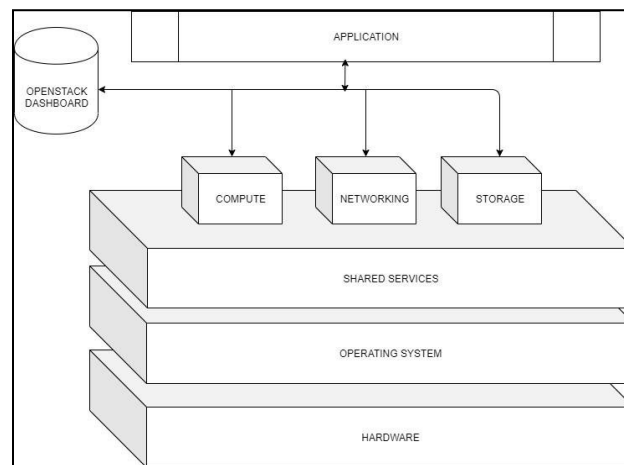


Fig 3. *OpenStack architecture* [source](#)

A cloud operating system called OpenStack manages enormous pools of computing, storage, and networking resources spread throughout a data center. These resources are all provided and managed using APIs using standard authentication techniques. Additionally, a dashboard is offered, allowing administrators command while enabling users to provision resources via a web interface. Beyond standard infrastructure-as-a-service functionality, additional components provide orchestration, fault management, and service management among other services to ensure the high availability of user applications[4]. In the process of using OpenStack, we used VMware for accessing the Linux OS(Ubuntu). Talking about the Cyber attack part of the project, we implemented a network-based cyber attack from one node to another and tried to detect it using snort, which is the most popular Open Source Intrusion Prevention System (IPS) in the world. In order to identify malicious network activity, Snort IPS employs a set of rules. It then searches for packets that fit those criteria and informs users when they do. Snort is the name of the most widely used Open Source Intrusion Prevention System (IPS) in the world. [5].

3. Implementation

For the implementation to begin we need to download and configure all the software and tools required for the project. In the previous sections, we have mentioned multiple tools that we have come across and implemented on the way. Let us talk about the first part of the setup. For this project, we had systems working with the windows operating system, and for using the Linux operating system we decided to use a VMware workstation. Specifically, we used the Ubuntu operating system for using the cloud platform named OpenStack. The reason for using VMware over Dual boot was the flexibility in the usage. Using a virtual machine gives the freedom to switch from host machine to virtual machine easily, and helps in working on both OS simultaneously. The fact that a virtual machine is sandboxed over dual-booting is one of its main advantages. This indicates that the virtualized operating system operates in a totally separate setting. The native operating system cannot be significantly impacted by anything in the virtualized operating system.

Another fantastic benefit of virtual machines is that it is not feasible with a dual-boot configuration. A complete snapshot of the entire operating system can be made, saved as a single file, and then transferred to another computer where it can be started as a virtual machine. You effectively copied the virtual OS. This helps in keeping track of every checkpoint and leads to backtracking any error that occurs. Clones are excellent for portability because you don't need the original physical machine to take your system anywhere. Snapshots, a related feature, are helpful for swiftly erasing changes. It will help you to restore the previous state of the machine in case something goes wrong. They also let you pause your work exactly as it was and resume it later.

A virtual machine is typically ideal for this if you want to utilize two distinct operating systems and need to transfer files between them or access the same files on both OS. Most virtualization software allows you to copy and paste between the host and virtual OS and easily set up shared folders that both can access. Virtualization software, like VMware, and VirtualBox, often offers an option called dynamically allocated storage for its virtual disks. You can specify the maximum disc capacity with this, but it will only use space when you add files to it. Take the case when you only want the VM to have 100GB, for instance. You choose dynamic storage and set the maximum size to 100GB. The VM's disc will only occupy 20GB of space on your actual storage drive while the OS installation and a few apps are running, unless you save more data to it.

Once we have installed the VMware workstation (we will be using the free version) we start preparing our virtual machine on it.

Virtual machine system configuration:

- **12Gb RAM**
- **100Gb Storage**
- **Ubuntu 20.04.4**

Once the machine is set and ready, we start the machine. As said before, the machine has Ubuntu 20.04.4 VM installed on it. Using the command line of the OS we will update all the existing libraries. The corresponding command for the same is “Sudo apt-get update”.

Now we start with the OpenStack installation for which we will be using Devstack to install OpenStack[6]. The OpenStack version used for the project is “YOGA”. The reason for using this particular Linux OS and version was the recommendation from the documentation and it's being tested widely with the particular version.

There are multiple ways to install OpenStack like MicroStack, DevStack, etc. The reasons for using DevStack are as follow:

DevStack is built from modern sources, and is purposed to use exclusively for testing and contributing to the OpenStack technologies themselves:

- "DevStack is a collection of extensible scripts based on the most recent versions of everything from git master," according to the website.
- DevStack alters your system's setup significantly, so it shouldn't be used in your primary development environment.
- DevStack "makes several decisions that are inappropriate for production systems," according to the author.

MicroStack is a well-designed installable package that can be used to prototype OpenStack-based systems and even deployed to production environments. According to the company, "MicroStack is OpenStack in a snap," which means that all services and auxiliary libraries are contained in a single package that is easily installed, upgraded, or removed.

So, we projected with lots of playing around with configurations and library versions, we decided to use DevStack and make things flexible on the go.

We need to set up the configuration for OpenStack before using it. The following configuration steps are:

1. ***Add a new user(Stack).***
2. ***Provide SUDO privileges to the Stack user.***
3. ***Create a local.conf file and add passwords for the devstack root.***

we projected with lots of playing around with configurations and library versions, we decided to use DevStack and make things flexible on the go.

Once the OpenStack is installed, we can access the application through a web browser using the admin credentials and entering into the dashboard, where one can find all the services and information provided by the DevStack Application.

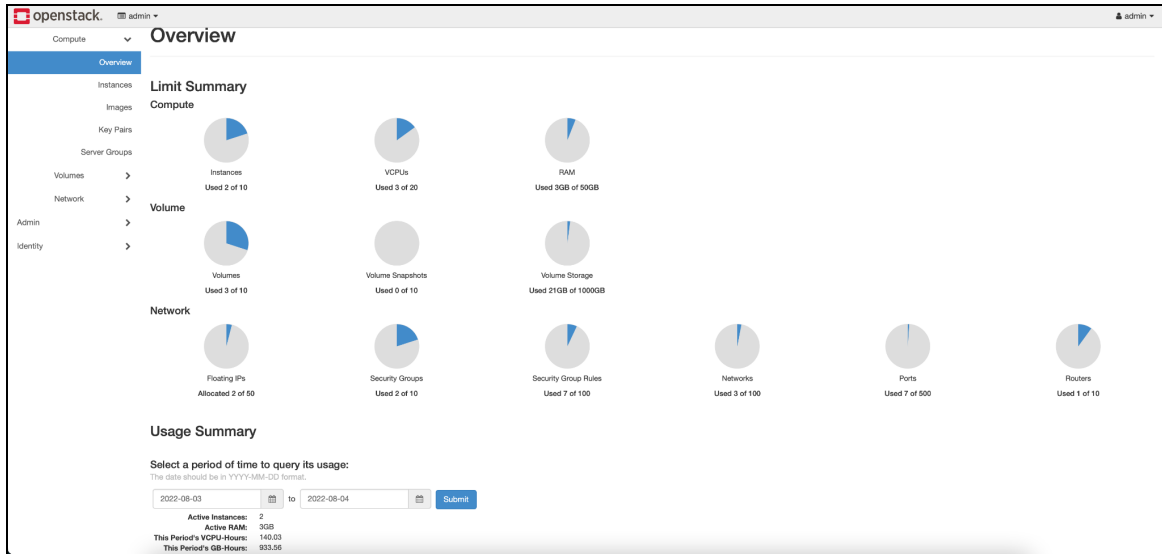


Fig 4. Horizon Dashboard

On the horizon Dashboard, we have various we have primary tabs,

- Admin
- Project
- Identity
- Settings

Once we have access to the OpenStack dashboard, we have the option to manage and upload images[11]. A virtual machine image, also known as an image in this document, is a single file that houses a virtual disc with an installed bootable operating system on it. In the cloud, virtual machine instances are created using images. Through the section, we uploaded the image file of the mentioned Ubuntu version. Once installed, it is easy to use the images for multiple instances created.

Let's talk about Network topology. We created the whole topology from scratch in the Networks consisted of.

- **Public Networks - 172.24.4.0/24**
- **Private 2 Networks**
 - **10.0.0.0/24 - Private 1**
 - **10.1.1.0/24 - Private 2**

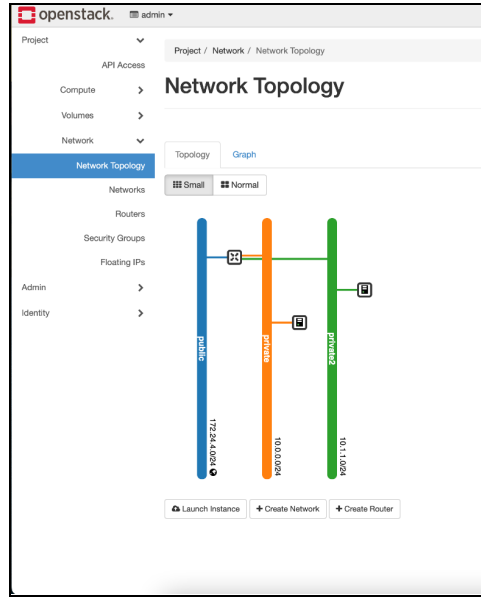


Fig 5. Network Topology

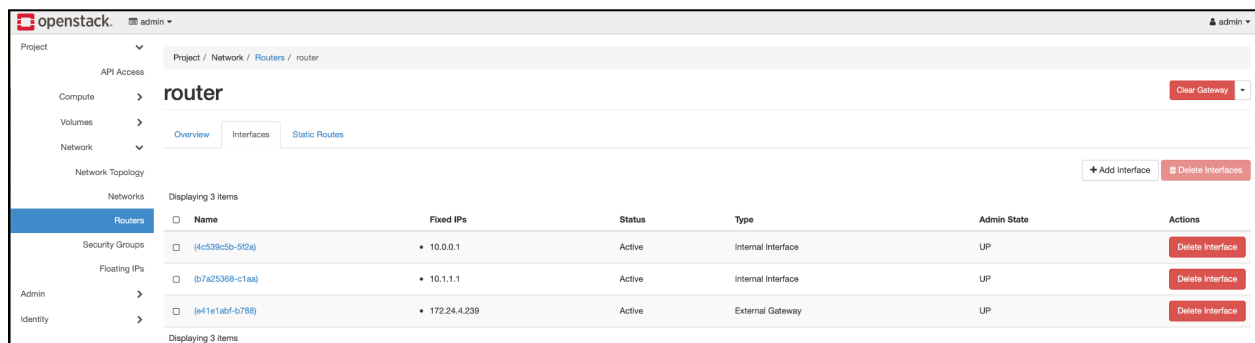
We as well set the parameters for the network, like...

- ***Network Name***
- ***Shared for Public Network***
- ***Non-Shared for Private Network***
- ***Provide Subnet Name***
- ***IP version (we have used IPV4)***

Once all network parameters are set we need to create a linking point between them so that the communication can take place. For this purpose, we will use routers and make the networks communicate. For router creation, the parameters need to be set similar to the Network parameters configuration.

The parameters go as follows:

- ***Router Name***
- ***Network type used - Public***
- ***Modify/Add interfaces of Private & Public networks.***



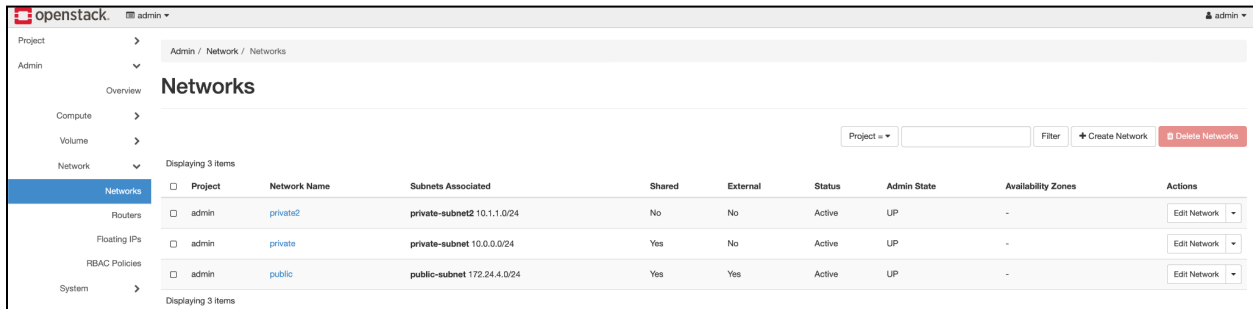
Name	Fixed IPs	Status	Type	Admin State	Actions
(4c53c5b-5f2a)	10.0.0.1	Active	Internal Interface	UP	Delete Interface
(b7a25368-c1aa)	10.1.1.1	Active	Internal Interface	UP	Delete Interface
(e41e1abf-b788)	172.24.4.239	Active	External Gateway	UP	Delete Interface

Fig 6. Router interfaces

The topology's configuration has now been finished up in its entirety. The next step in the process is "instance creation," which is very crucial. From a public or private cloud network a virtual server instance is referred to as a "cloud instance". A single piece of hardware is converted into software and used to power several computers in cloud computing instances. Customers don't have to worry about how many servers can run on a specific piece of hardware without noticeably slowing down the application during high usage because cloud instance computing is so flexible. You can simply add extra machines if performance reaches its maximum. By using cloud instance computing, the cloud software may easily be expanded to span many machines, either permanently or temporarily, in the event that the server develops beyond the capabilities of a single machine. Additionally, the downtime associated with hardware maintenance is decreased by cloud instance computing.. A server in the cloud can be easily moved from one physical machine to another without going down. The cloud's abstraction makes it possible for hardware to effortlessly transport all data between locations without the end-user being aware of it. In sum, cloud instance computing is highly dynamic, can reassign resources as needed, and allows for the movement of servers as they run in the cloud[7].

For the Project we have created 2 instances, one acts as an attacker, and the other acts as a victim. We have provided the same resources for both instances. The instances have a storage capacity of 10GB and 2GB RAM. The other important parameters related to instances are:

- ***Provide Instance Name***
- ***Default availability zone NOVA***
- ***Generate SSH key pair***
- ***Network - Public/Private***



Project	Network Name	Subnets Associated	Shared	External	Status	Admin State	Availability Zones	Actions
admin	private2	private-subnet2 10.1.1.0/24	No	No	Active	UP	-	Edit Network
admin	private	private-subnet 10.0.0.0/24	Yes	No	Active	UP	-	Edit Network
admin	public	public-subnet 172.24.4.0/24	Yes	Yes	Active	UP	-	Edit Network

Fig 7. Instances in the network

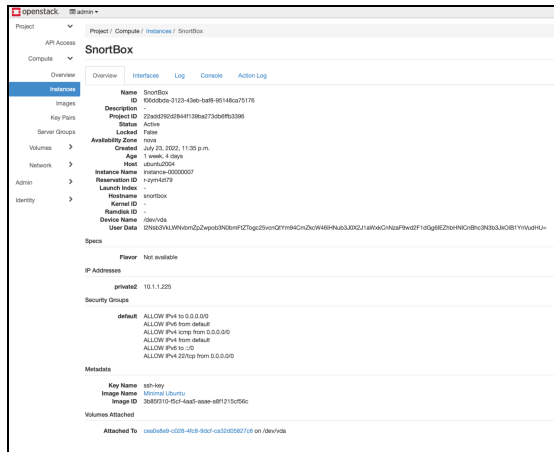


Fig 8. Instance 1 details

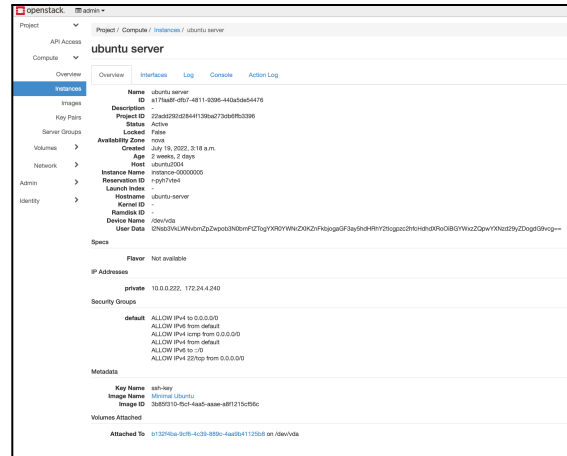


Fig 9. Instance 2 details

4. Attacker and Victim Instance

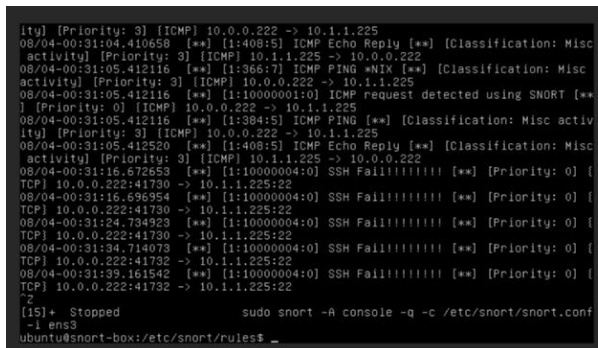


Fig 10. Victim Instance terminal view

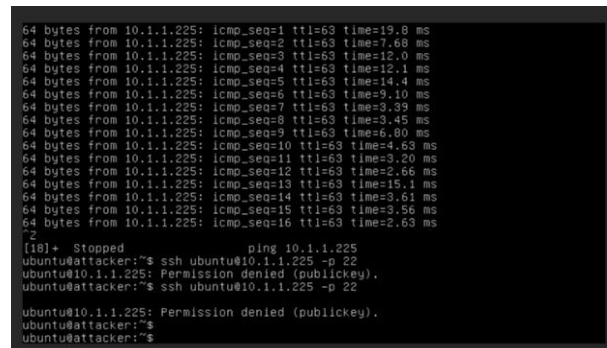


Fig 11. Attacker Instance terminal view

In this section we will talk about the attacker and the victim node. We have created two instances each of them acting as an attacker and victim. Each is configured accordingly. First, we will talk about Attacker Node, Attacker node as the name suggests has been assigned the role to conduct an attack on other nodes and the same goes for the victim node.

4.1. SNORT Introduction

Snort IPS searches for packets that match against a set of rules it has created to help define malicious network activity. It then provides alerts for users. To identify malicious activities, such as denial-of-service (DoS) assaults, buffer overflows, stealth port scans, CGI attacks, SMB probes, and OS fingerprinting efforts, it employs a rule-based language that combines signature, protocol, and anomaly inspection methods. On IP networks, it is capable of carrying out real-time traffic analysis and packet logging.[5]

Additionally, Snort can be implemented inline to block these packets. Snort has three main applications It can be used as a packet sniffer, similar to tcpdump, as a packet logger, useful for debugging network traffic, or as a full-fledged network intrusion prevention system. Both private and commercial users can download and set up Snort.

Snort can be set up in three different ways: Snort can be configured in three main modes:

1. Sniffer Mode
2. Packet logger Mode and
3. Network Intrusion detection System Mode.

- **Sniffer Mode**

Network packets will be read by the software and shown on the console.

- **Packet Logger Mode**

The application will log packets to the disc while in packet logger mode.

- **Network Intrusion Detection System Mode**

When in intrusion detection mode, the application will track network activity and examine it in comparison to a user-defined set of rules. Based on what has been identified, the computer will subsequently carry out a certain action.

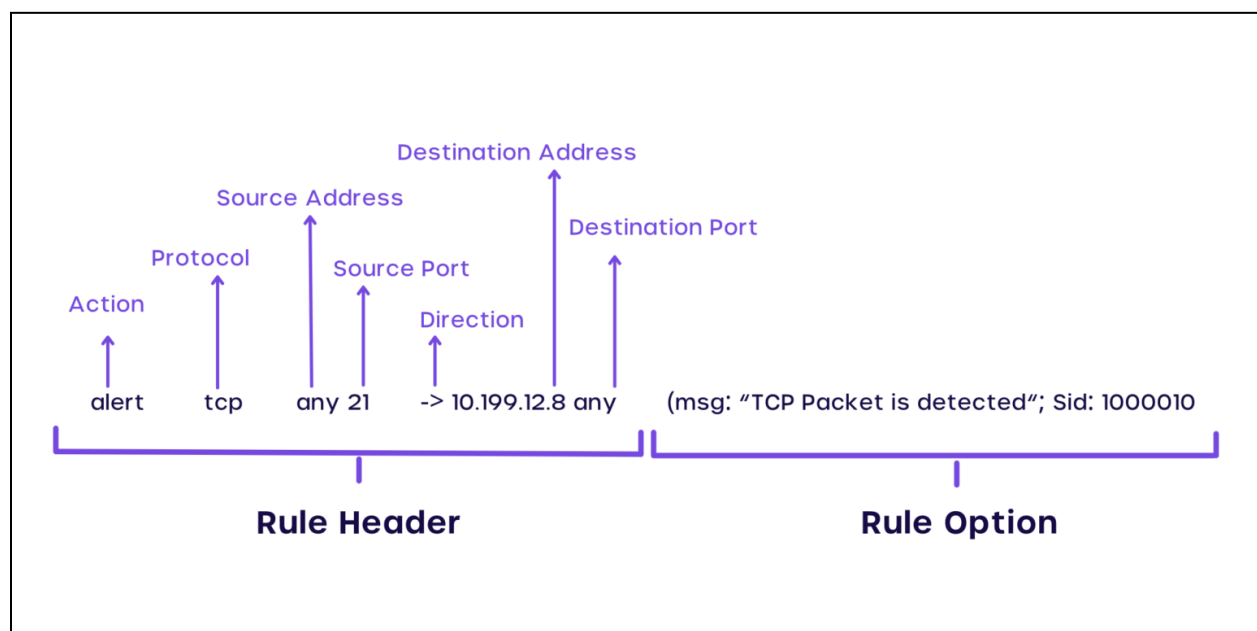


Fig 12. *Snort rules explained*[8]

a. Rule Header

1. **Rule Action:** There are 5 rule actions by default while you run a typical Snort rule: Alert, Dynamic, Pass, Log, or/and Activate. The most typical rule action is "alert," which logically alerts the network administrator when a potential threat is discovered.
2. **Protocol:** As was already said, a computer's protocol is its individual address. You can provide the Protocol address that has to be avoided based on past incidents or historical data while designing the rule.
3. **Source IP Address:** Assume that Mr. ABC is the source of your danger. The source IP address is sometimes his network ID or his IP address. You can just include the word "any" in this section of the rule if you need to receive alerts from all sources.
4. **Source Port:** This functions as a kind of facilitator for message transmission between two or more computer networks. You can type "any" one more time to define all 65,536 TCP Ports that are typically found on computers in the rule.
5. **Flow:** The flow of traffic is referred to in Snort Rules. This term or symbol enables us to apply rules selectively to particular segments of the flow. The symbol for it is an arrow (→).
6. **Destination IP Address:** Where do the network packets from the Source IP address channeled through Source Port and Destination Port go? No prizes for guesses. They go to the Destination IP Address. So obvious, isn't it?
7. **Destination Port:** Just like the source port which facilitates communication between the source ports, the Destination port does the same with Destination TCP/ UDP.

b. Options

Popular 'options' include Content, Offset, Content-List, Flags, etc. Each of these choices, which are entered near the conclusion of the rule line, substantially describes the rule's purpose and the results it produces.

4.2. Snort Installation and Configuration

Step 1: To create a victim machine, we need to install Snort

```
apt-get update
```

```
sudo apt install snort
```

Step 2: Configuration

Configure the interface as default is set to eth0. Need to change the interface as per the network connections. (In our case it is ens3)

Next we need to configure the network range and apply it. In our case it is 10.1.1.0/24

After this it will start the Snort installation. To verify the installation we use below command,

snort -V

Step 3: Custom Rule Writing

We create a file with .rules extension in /etc/snort/rules/ folder.

Once the file is saved we need to load this rule in snort.conf file for which we use below command,

sudo snort -T -i <interface (eth0)> -c <new configuration file location>

Ex: **sudo snort -T -i ens3 -c /etc/snort/snort.conf**

Step 4: To check if the rule works we perform below commands,

sudo snort -A console -q -i <Interface> -c <new configuration file location>

Ex: **sudo snort -A console -q -i ens3 -c /etc/snort/snort.conf**

Executing this command will run the Snort

4.3. ICMP Attack

We launched a PING Flood attack. Ping flood, often referred to as ICMP flood, is a typical Denial of Service (DoS) assault in which the attacker overwhelms the victim's computer with ICMP echo requests, also known as pings, to bring it to a standstill[9].

With the knowledge that the victim's network will respond with an equivalent amount of reply packets, the assault includes saturating it with request packets. This uses up a lot of bandwidth and results in a denial of service on the network's incoming and outgoing channels. Normally, ping requests are used to test the connectivity of two computers by measuring the round-trip time from when an ICMP echo request is sent to when an ICMP echo reply is received. During an attack, however, they have used to overload a target network with data packets.

The IP address of the target must be known by the attackers in order to carry out a ping flood. Based on the target and how its IP address is resolved, attacks can thus be divided into three groups.

- A single computer on a local network is the target of a targeted local revealed ping flood. The computer must be physically accessible for an attacker to learn its IP address. If the assault is successful, the target machine will be shut down.
- A router disclosed ping flood targets routers in order to disrupt communications between computers on a network. It depends on the attacker being aware of a local router's internal IP address. In the event of a successful attack, the router would be shut off along with any linked PCs.
- Before launching an assault, a blind ping flood uses an external software to find out the IP address of the target network or computer.

We used PING on the command line, to ping the victim instance and flood it with multiple PING commands

- *Ping 10.1.1.225*

Snort rule used for ICMP Attack

etc/snort/rules -> icmp.rules

alert icmp any any -> 10.1.1.225 any (msg: "ICMP request detected using snort"; sid:10000001;)

4.4. SSH failed login detection using Snort.

We have performed an attack where every failed login on the victim machine through SSH will be detected on Snort.

Secure Shell (SSH) is a network protocol used to securely access the device over an unencrypted network. It is mainly used by network administrators to manage a system/application, execute commands remotely and transfer files. SSH server listens on a standard Transmission Control Protocol (TCP) port 22, by default.

The attacker tries to login on the victim machine with SSH. A Host Based Intrusion Detection, Snort is used to detect the failed login. We have created a custom rule which will generate an alert whenever a failed login attempt occurs on the victim machine, as follows,

Snort rule used in the project:

alert tcp 10.0.0.225 any -> 10.1.1.222 22 (msg:"SSH Fail !!!!!"; count 1; sid:10000024; rev: 1;)

The above rule is created for a TCP protocol, with the source address of the victim machine (10.0.0.225) to the destination address 10.1.1.222 on SSH default port 22. A message (msg) parameter is used to set the alert text, notifying the user. Count parameter is used to set the number of failed login attempts. SID parameter is used to uniquely identify the snort rule, and rev parameter is used to uniquely identify revision of the Snort rule.

5. Obstacles

While in the process of implementing the project we came across multiple obstacles and In this section, we would be mentioning them and how we overcame them to move ahead in the process.

1. While installing Openstack we had issues with multiple python libraries as their version was conflicting with the Openstack installation. Using the latest version of ubuntu had the latest versions of all the copies incompatible with the OpenStack version, and using a lower version of ubuntu had downgraded versions of the python library.
2. Function- common timeout error, causing the installation process of OpenStack. CPU resources used by the timed-out function instance are throttled and request processing is paused. The error was solved by increasing the time duration for the timeout.

- a. Set a timeout longer than the time you anticipate it will take your function to complete.
 - b. Track the amount of time left during execution and perform cleanup/exit early.
3. The next big obstacle we faced was dealing with the existing system configuration. We had systems with less RAM and Memory leading to exhaustion of resources while using OpenStack, especially while Virtual machine Creation and instance creation.
4. Every time we stopped the virtual machine, we faced the issue with Openstack as it halted abruptly, and on the attempt of logging in we could not use the Openstack service available on the dashboard. Then we started taking *snapshots* of the process before the abrupt halt and traced back to start from where it halted.
5. We were unable to create an “extnet” for the public network. We solved it by configuring the controller node, through the */etc/neutron/plugin*.
6. Ubuntu server password injection, as we were not able to login into ubuntu cloud image and also ssh was not working properly, so we did admin password injection, */etc/nova/nova.conf*

#cloud-config

hostname: snort-box

fqdn: snort_build

ssh_pwauth: False

password: ubuntu
7. Because of only 4 cores we were only able to run 2 instances on top of OpenStack, if we launch 1 more instance then another instance would start to shut off. We solved this problem by trial and error in giving resources while instance creation.
8. For installing dependencies, we needed to connect to the internet, which was a tedious task, so we changed our subnet and added 8.8.8.8 into the DNS config.

6. Conclusion

In this project, we came across multiple tools, and went through various steps to install and implement the tools according to the requirements of the process. We came across openstack installation and created multiple instances and conducted security based activities in each of them. We grasped the process and how cloud security works in each scenario. The project had a major part in the installation process which made us know the importance of hardware required in IaaS based cloud. The utilization of each service in a cloud and resource management has been imbibed during the project. We successfully implemented two attacks and detected them within two nodes.

7. Contributions

- **Install VMware and Deploy OpenStack and Learn about its architecture**
 - Himanshu Mahajan
 - Harish Krishna Venkateswaran
- **Designing Virtual Network in Horizon dashboard**
 - Mehul Patil
 - Himanshu Mahajan
- **Instance Creation and setup**
 - Varnesh Gawde
 - Sushant Padmanabhi
- **Performing attack in the created network**
 - Harish Krishna Venkateswaran
 - Mehul Patil
- **Deploying network security tools and detecting the conducted attacks**
 - Varnesh Gawde
 - Sushant Padmanabhi
- **Collection of resources and Report Creation**
 - Vasu Gandhi
 - Varnesh Gawde
 - Harish Krishna
 - Sushant Padmanabhi
 - Himanshu Mahajan
 - Mehul Patil
 - Tanmya Rampal
- **Live Demonstration and Presentation**
 - Harish Krishna Venkateswaran
 - Himanshu Mahajan
 - Varnesh Gawde
 - Sushant Padmanabhi

References

1. P. Mell and T. Grance, "Draft NIST working definition of cloud computing - v15", vol. 21, Aug 2009 2009.
2. Y. Chen, V. Paxson and R. Katz, "What's New About Cloud Computing Security?", 2010.
3. Chen Donglin, Fu Min, Chen Ling, et al. "Research on the Construction Mode of Experimental Teaching Platform Based on Hybrid Cloud [J]", *Journal of Experimental Technology and Management*, (5), (2013).
4. <https://www.openstack.org/software/>
5. <https://www.snort.org/>
6. <https://docs.openstack.org/devstack/latest/>
7. <https://apprenda.com/library/glossary/definition-cloud-instance-single-multi>
8. <https://cyvatar.ai/write-configure-snort-rules/>
9. <https://www.imperva.com/learn/ddos/ping-icmp-flood/>
10. <http://manual-snort-org.s3-website-us-east-1.amazonaws.com/>
11. <https://docs.openstack.org/horizon/latest/user/manage-images.html>
12. <https://docs.openstack.org/mitaka/networking-guide/>