

**Project final report**

**INSE 6620: Cloud Computing Security and Privacy**

**Submitted to**

**Lingyu Wang**



**GINA CODY**  
SCHOOL OF ENGINEERING  
AND COMPUTER SCIENCE

**Submitted by**

Omer Mujtaba

Simon Benoit

Sami Islam

Daniel Kabagema

James Kearney

Pooja Singh

Saeid Jamshidi

Ho-Yang Chang

**August 11, 2020**

# Table of Content

Introduction.....	1
OpenStack.....	1
Installation .....	1
Configuration-Defining Network .....	2
Configuration-Router Configure .....	2
Configuration-Security Group.....	3
Configuration-Networking .....	3
Configuration-Floating Ips .....	4
Configuration-Upload Image.....	4
Configuration-Create Instances .....	4
Outcome .....	5
Snort.....	6
Installation .....	6
Configuration-Configure Snort to run in NIDS mode .....	6
Configuration-Snort Rules.....	6
Outcome .....	7
Splunk .....	7
Installation.....	7
Configuration-Splunk Receiver .....	8
Configuration-Splunk Forwarder .....	8
Outcome .....	9
LAMP server .....	9
Installation .....	9
Attack Methodology .....	10
DoS attack-hping3:.....	10
SQL injection-sqlmap:.....	10
Challenges .....	11
Deploying OpenStack.....	11
Solved Problem: Install scripts error.....	11
Solution examples: .....	11
Unsolved Problem: No external access of Instances .....	12
Attempt to solve: .....	12
Unsolved Problem: OVS bridging .....	13
Platform issue: Google cloud .....	13
Building up virtual bridge.....	14
Summary .....	15
Contribution table(tasks) .....	I
Installation Script with Packstack .....	II
Installation Script of Snort .....	IV
Snort rules set for our cloud.....	IV
References .....	V

## Introduction

OpenStack is a set of tools that allows you to build and manage private and public clouds. The goal of this project is to give the project stakeholders exposure to working within a team to build a cloud environment from the ground up. Providing members with an understanding of what happens underneath the veil and solidifying all the concepts learned in class. The work for this project was broken into three areas: building an OpenStack cloud environment, designing a virtual network with reporting tools, and using the tools to defend the network from an attack.

This document will showcase the details and the work that went into implementing our OpenStack network. We will enumerate the processes used to install OpenStack as well as highlight the many challenges we faced along the way, and the steps we took to resolve them. This paper will demonstrate the hard work and effort put forth by the team to ensure that the project outcome was in line with our desired result.

This report will go over the OpenStack installation and configuration. The tools implementation including Snort, Splunk, and a LAMP server. The attack methodology to test the installed tools. Finally, we present the challenges, describing in detail what we have overcome in order to enable our team to have a functional OpenStack network.

## OpenStack

This section describes the detailed installation and configuration process of OpenStack, also the detailed specification about OS installed on a virtual machine, services packet, etc. OpenStack provides the services as an infrastructure for virtual machines.

## Installation

Based on a single entry in a community forum, we decided to try a dual interface installation. One "Host Only Interface" and one bridged interface. The host only interface would be the OpenStack install interface and the bridge interface would be dedicated to Internet access. Thus, removing the bridge issue completely and using layer 3 networking to exit internet-bound traffic.

We chose to install OpenStack on Ubuntu 18.04 LTS as this was the distribution OpenStack claimed to be the most tested. After the changes we made to our physical setup, the installation went quite smoothly with little trouble.

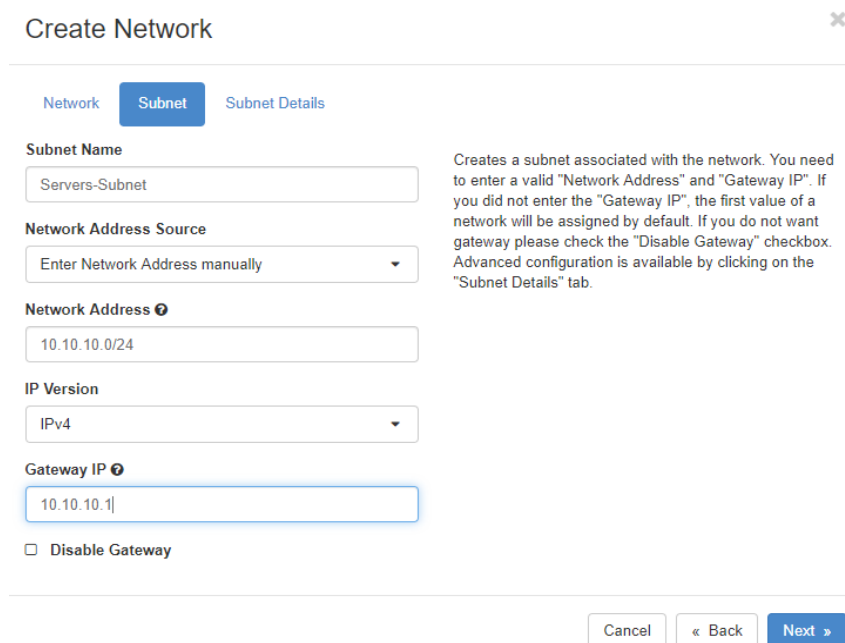
1. Install Ubuntu 18.04 LTS with minimal options and SSH server
2. Fixing the known issues, we would encounter that would halt install or cause problems with OpenStack functionality.
  - i. Disable ufw
  - ii. Install Python3-OpenStack client
  - iii. Download the current version of Devstack
  - iv. Fix the error in outfilter.py
  - v. Manually upgrade wrapt and SimpleJson
3. created a very simple local.conf that only contained the passwords for the services, which allows the OpenStack installer to handle the rest of the set-up easily.
4. Run ./stack.sh to start the install

OpenStack has been successfully deployed with the setup of the following services were built:

- Cinder – Block storage service
- Neutron – Networking service
- Nova – Compute
- Swift – Object storage service
- Keystone – Identity Service
- Heat – Orchestration Service
- Glance – image service
- Horizon – Dashboard
- Magnum – Container service

## Configuration-Defining Network

We followed the network structure submitted in the proposal. Public (internet) , Private (Tenant), Server (our Servers), Users (our Users). The process was quite simple, after the installation, log in to the Horizon interface using the credentials defined in the local.conf file during the installation process. A basic network and router were already installed in the admin context, just needed to be ensured they were made public and accessible to the Demo project. After that, we were able to add additional networks, and made sure that the gateway and addressing were all set correctly.



The screenshot shows the 'Create Network' form in the OpenStack Horizon interface, specifically the 'Subnet' tab. The form includes the following fields and options:

- Subnet Name:** A text input field containing 'Servers-Subnet'.
- Network Address Source:** A dropdown menu set to 'Enter Network Address manually'.
- Network Address:** A text input field containing '10.10.10.0/24'.
- IP Version:** A dropdown menu set to 'IPv4'.
- Gateway IP:** A text input field containing '10.10.10.1'.
- Disable Gateway:** An unchecked checkbox.

On the right side of the form, there is a help text block: 'Creates a subnet associated with the network. You need to enter a valid "Network Address" and "Gateway IP". If you did not enter the "Gateway IP", the first value of a network will be assigned by default. If you do not want gateway please check the "Disable Gateway" checkbox. Advanced configuration is available by clicking on the "Subnet Details" tab.'

At the bottom right, there are three buttons: 'Cancel', '« Back', and 'Next »'.

Figure 1. Horizontal Interface

## Configuration-Router Configure

Creating an interface on the router to compose the network with it, includes one interface for each of the networks ensuring that the address assigned to that interface is the gateway address of the network. This ensures that the default routing works as intended.

Add Interface

Subnet \*

Select Subnet

Select Subnet

shared: 192.168.233.0/24 (shared-subnet)

Description:

You can connect a specified subnet to the router.

If you don't specify an IP address here, the gateway's IP address of the selected subnet will be used as the IP address of the newly created interface of the router. If the gateway's IP address is in use, you must use a different address which belongs to the selected subnet.

Cancel

Submit

Figure 2. Adding interface on router

## Configuration-Security Group

The security groups essentially act as the firewall to the OpenStack. It allows us to permit or deny access to the instances with fine-grained rules. Since implementing a firewall in our network is determined, we opted to allow all traffic. The installed firewall will be the only publicly accessible instance, all others are routed through the firewall instance.

## Configuration-Networking

All the related network changes done in the OpenStack interface make the necessary changes in Neutron module. Neutron modules control all of the Network layer virtualization, controlling the setup of virtual tap interfaces to the host for each of the instance interfaces. It also controls the iptable entries on the host for the virtual network, which is obfuscated through the security groups tab in the Horizon interface.

Figure below illustrates the iptable entries that reflect the security groups changes we made. The highlighted are the network address translation rules that are required to provide the firewall instance access to the external bridge with a mapped IP address.

```
stack@openstack: ~/devstack$ sudo ip netns exec qrouter-6e872417-bf86-4bb2-95d2-5eb55a1c600f iptables -t nat -xnvL
Chain PREROUTING (policy ACCEPT 1551 packets, 109580 bytes)
 pkts bytes target prot opt in out source destination
 1786 126281 neutron-l3-agent-PREROUTING all -- * * 0.0.0.0/0 0.0.0.0/0

Chain INPUT (policy ACCEPT 85 packets, 6162 bytes)
 pkts bytes target prot opt in out source destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target prot opt in out source destination
 0 0 neutron-l3-agent-OUTPUT all -- * * 0.0.0.0/0 0.0.0.0/0

Chain POSTROUTING (policy ACCEPT 3 packets, 204 bytes)
 pkts bytes target prot opt in out source destination
 1641 115419 neutron-l3-agent-POSTROUTING all -- * * 0.0.0.0/0 0.0.0.0/0
 1261 95411 neutron-postrouting-bottom all -- * * 0.0.0.0/0 0.0.0.0/0

Chain neutron-l3-agent-OUTPUT (1 references)
 pkts bytes target prot opt in out source destination
 0 0 DNAT all -- * * 0.0.0.0/0 172.24.4.142 to:10.0.0.36

Chain neutron-l3-agent-POSTROUTING (1 references)
 pkts bytes target prot opt in out source destination
 380 20008 ACCEPT all -- * !qg-15ald6cb-6a 0.0.0.0/0 0.0.0.0/0 ! ctstate DNAT

Chain neutron-l3-agent-PREROUTING (1 references)
 pkts bytes target prot opt in out source destination
 135 8100 REDIRECT tcp -- -- -- * 0.0.0.0/0 169.254.169.254 tcp dpt:80 redir ports 9697
 3 204 DNAT all -- * * 0.0.0.0/0 172.24.4.142 to:10.0.0.36

Chain neutron-l3-agent-float-snat (1 references)
 pkts bytes target prot opt in out source destination
 270 20549 SNAT all -- * * 10.0.0.36 0.0.0.0/0 to:172.24.4.142
```

Figure 3. iptable entries

## Configuration-Floating Ips

The last piece of networking we needed to configure is the Floating ips. These allow us to assign individual instances IP addresses that can be routed through the host, allowing instances to be accessed from external users.

## Configuration-Upload Image

The Nova service controls all aspects of compute in OpenStack, including the instances we create and the hypervisor, which has a direct effect on the image types we use. By default, OpenStack uses QEMU as the hypervisor and it is recommended to use Qcow2 cloud images on this hypervisor. For our network we chose to use Ubuntu again and uploaded **bionic-server-cloudimg-amd64.img** (qcow2 cloud-optimized image).

## Configuration-Create Instances

With the networking in place we created 4 instances. A firewall, with connections to all of our networks and assigned a floating IP to provide dedicated external access. We also created two servers and a user workstation, each on its appropriate network. When creating the instances, we provided them a name, an image to be used, a flavour (the size of the instance to be created), which networks would be assigned (OpenStack assigns an IP address via dhcp) and finally a key pair. The key pair is necessary as by default user passwords are not created and using SSH certificates is the only way to login.

Launch Instance

Details

Source

Flavor

Networks

Network Ports

Security Groups

Key Pair

Configuration

Server Groups

Scheduler Hints

Metadata

Networks provide the communication channels for instances in the cloud.

▼ Allocated 3

Select networks from those listed below.

	Network	Subnets Associated	Shared	Admin State	Status	
1	private	ipv6-private-subnet private-subnet	No	Up	Active	↓
2	Users	Users-subnet	No	Up	Active	↓
3	Servers	Servers-Subnet	No	Up	Active	↓

▼ Available 1

Select at least one network

Click here for filters or full text search.

	Network	Subnets Associated	Shared	Admin State	Status	
>	shared	shared-subnet	Yes	Up	Active	↑

Cancel

< Back

Next >

Launch Instance

Figure 4. Instance control through GUI

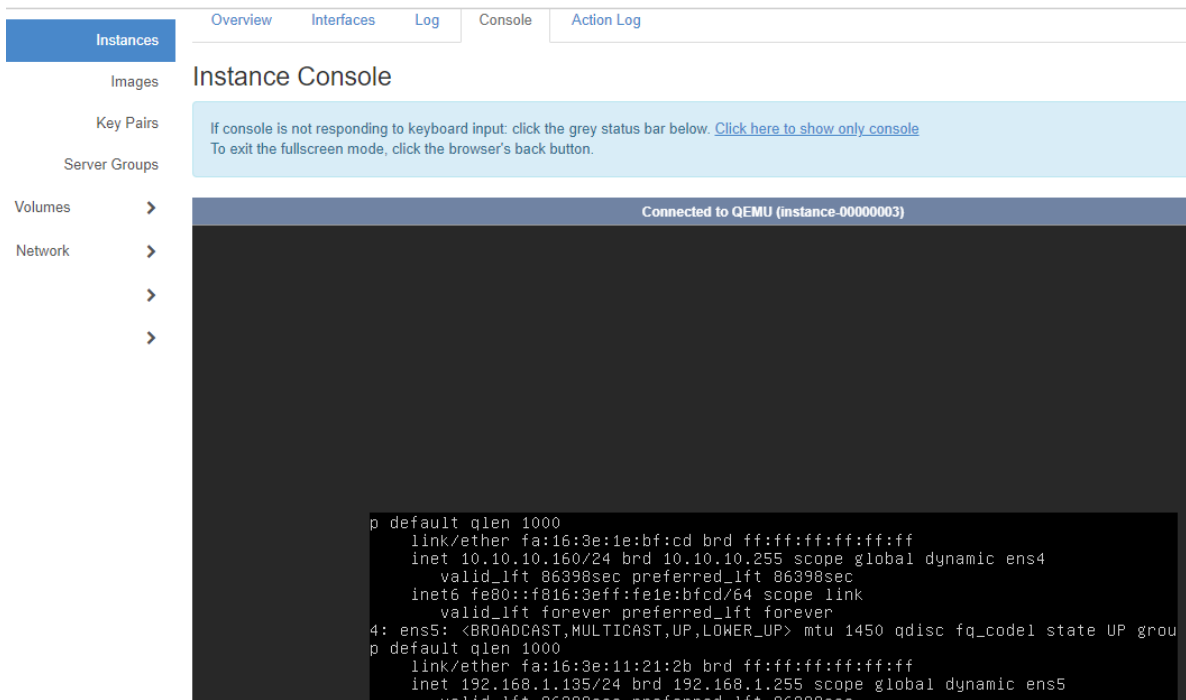


Figure 5. Console access through GUI

## Outcome

Now all the configuration is set, combining with security tools can be done further. Figure below shows the whole structure of the created network.

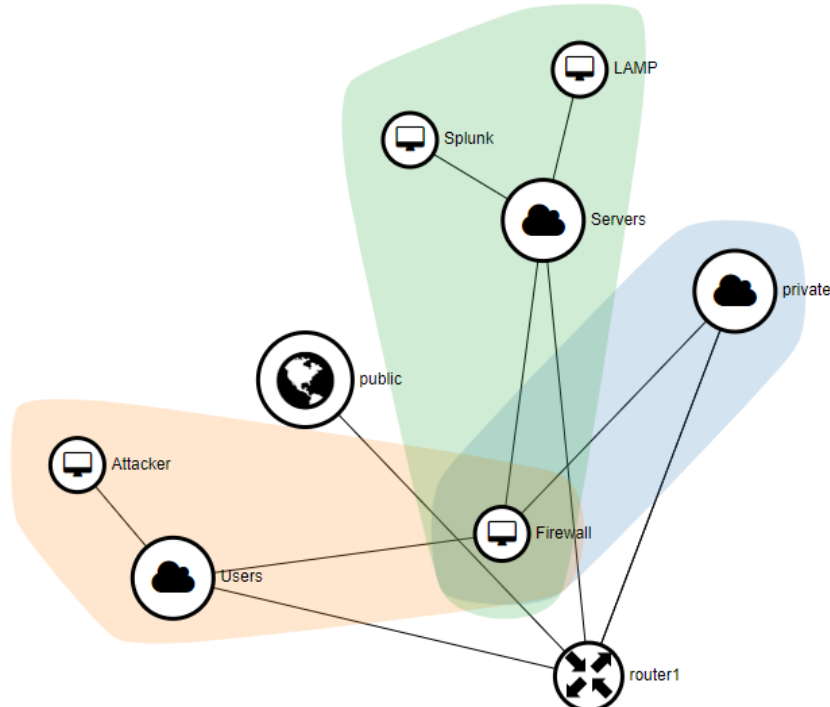


Figure 6. Final layout of Network

## Snort

This section provides the walk-through of the installation and configuration process of Snort. We have created a couple of scripts to run the snort installation, and some other scripts to run snort.

### Installation

Snort was installed on the firewall instance, this allowed us to set up alerts directly on this server. The scripts referred above is included in the Appendix.

1. *Install Prerequisites for Snort*
2. *Making a temporary download folder*
3. *Download the latest Data Acquisition library (DAQ) source package from the Snort website with the wget command underneath. Snort itself uses DAQ to make abstract calls to packet capture libraries.*
4. *When completed, the source code can be extracted and jump into the new directory with the following commands.*
5. *Configure the installation with sourcefire enabled, run make and make install.*

### Configuration-Configure Snort to run in NIDS mode

1. *Snort on Ubuntu safely without root access, you should create a new unprivileged user and a new user group for the daemon to run under.*  
`sudo groupadd snort`  
`sudo useradd snort -r -s /sbin/nologin -c SNORT_IDS -g snort`
2. *Then create the folder structure to house the Snort configuration, just copy over the commands below.*  
`sudo mkdir -p /etc/snort/rules`  
`sudo mkdir /var/log/snort`  
`sudo mkdir /usr/local/lib/snort_dynamicrules`
3. *Set the permissions for the new directories accordingly.*  
`sudo chmod -R 5775 /etc/snort`  
`sudo chmod -R 5775 /var/log/snort`  
`sudo chmod -R 5775 /usr/local/lib/snort_dynamicrules`  
`sudo chown -R snort:snort /etc/snort`  
`sudo chown -R snort:snort /var/log/snort`  
`sudo chown -R snort:snort /usr/local/lib/snort_dynamicrules`
4. *Create new files for the white and blacklists as well as the local rules..*  
`sudo touch /etc/snort/rules/white_list.rules`  
`sudo touch /etc/snort/rules/black_list.rules`  
`sudo touch /etc/snort/rules/local.rules`
5. *Then copy the configuration files from the download folder.*  
`sudo cp ~/snort_src/snort-2.9.16/etc/*.conf* /etc/snort`  
`sudo cp ~/snort_src/snort-2.9.16/etc/*.map /etc/snort`

### Configuration-Snort Rules

The snort rules were set up to detect attacks on the server. This section will only include two rules and the other rules will be included in the Appendix.

```
alert tcp any any -> any any ( msg: "SYN Flood"; flags:s; flow: stateless;
detection_filter: track by_dst, count 40, seconds 10; sid:10000001; rev:001;
classtype:attempted-dos;)
```





1. *Download Splunk Installer*  
`cd /tmp && wget`  
<https://download.splunk.com/products/splunk/releases/7.1.1/linux/splunk-7.1.1-8f0ead9ec3db-linux-2.6-amd64.deb>
2. *Install Splunk*  
`sudo dpkg -i splunk-7.1.1-8f0ead9ec3db-linux-2.6-amd64.deb`
3. *Enable the Splunk to start at boot*  
`sudo /opt/splunk/bin/splunk enable boot-start`
4. *Start the Splunk service*  
`sudo service splunk start`
5. *Start the Splunk service*  
`sudo service splunk start`
6. *Splunk will be started at port 8000. You can access the application via URL*  
<http://localhost:8000>

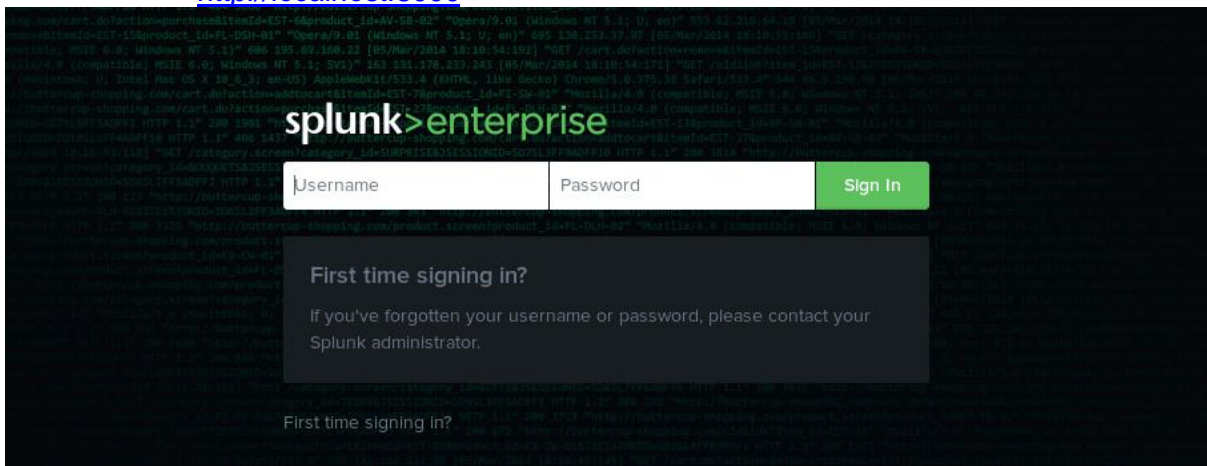


Figure 10. Login page of Splunk

## Configuration-Splunk Receiver

The purpose of Splunk Receiver is to configure Splunk to be able to receive data from forwarder.

1. *Login with the Splunk server user and pass you created previously*
2. *Go to Settings → Forwarding and Receiving (under data) → Configure Receiving (under receive data)*
3. *Click New Port and enter a port for the server to listen on (example port 9997) and click save*
4. *Go back to the terminal and run the command `netstat -auntp | grep port` to ensure its listening*

## Configuration-Splunk Forwarder

The purpose of Splunk Forwarder is to deploy among firewall or LAMP servers to pass data to Splunk.

1. *Download the Splunk Universal forwarder on the Splunk website*  
[https://www.splunk.com/en\\_us/download/universal-forwarder.html](https://www.splunk.com/en_us/download/universal-forwarder.html)
2. `tar -xvzf Splunk.tgz File Name`
3. `cd splunkforwarder/bin`
4. `./splunk start --accept-license`
5. *Set user and pass for Splunk forwarder*
6. `./splunk add forward-server <SplunkInstanceIP>:<port>`
7. `./splunk add monitor /var/log/auth.log -sourcetype linux_secure`
8. `./splunk add monitor /var/log/syslog -sourcetype syslog`

9. `./splunk restart`

## Outcome

After installation and configuration, Splunk was able to receive data and allow the user to search and detect threats or set alarms.

1. *Go to Search and Reporting*
2. *“What to search” should get populated with events.*

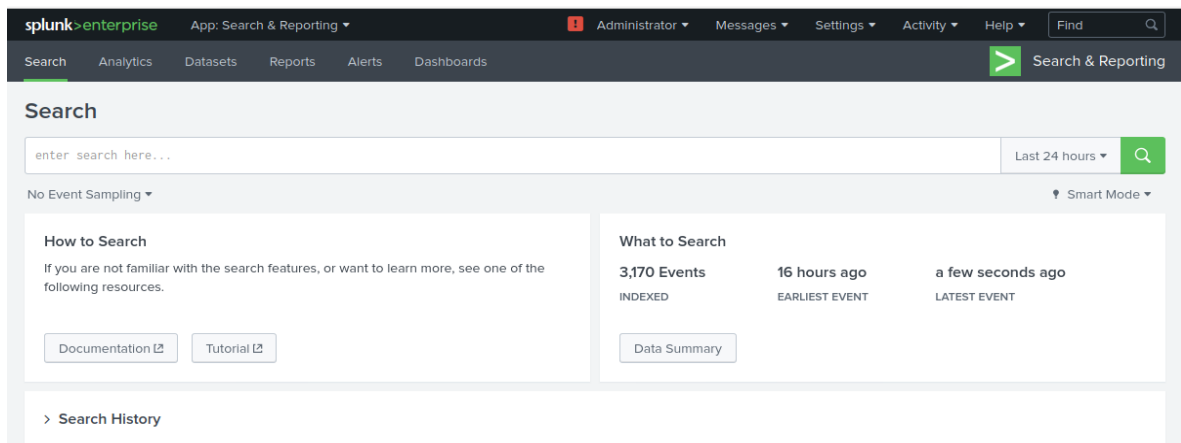


Figure 11. Interface of Splunk

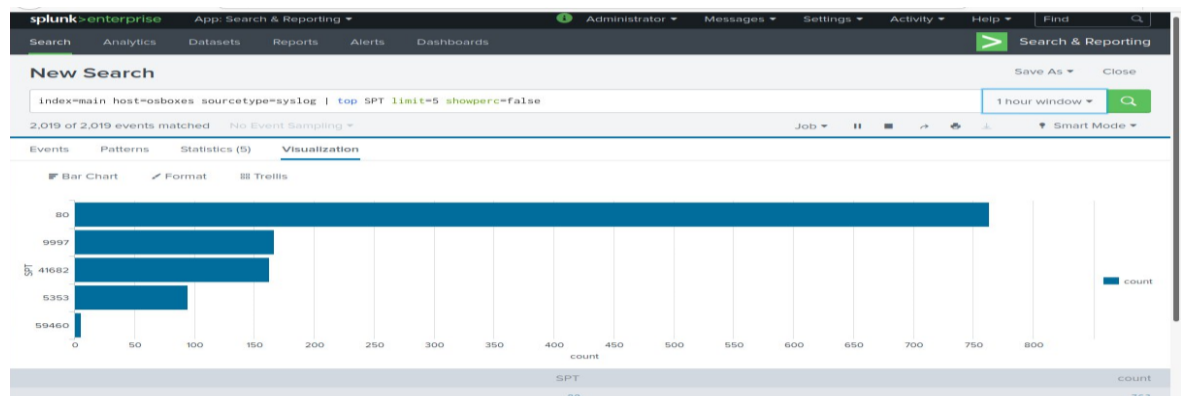


Figure 12. Dashboard of Splunk when SYN flood

## LAMP server

This section will illustrate the installation process of LAMP server (Linux operating system, with the Apache web server. The data is stored in a MySQL database, and dynamic content is processed by PHP).

### Installation

1. *Download package information from all configured sources.*  
`sudo apt get update`
2. *Install Apache*  
`sudo apt get apache2`
3. *Enable Firewall*  
`sudo ufw enable`
4. *Install MySQL server/client, php and phpmyadmin*  
`sudo install mysql-server mysql-client`

```

sudo install php7.4 libapache2-mod-php7.4
sudo apt install phpmyadmin
5. Restart apache2
sudo systemctl restart apache2.service

```

## Attack Methodology

The attack was expected to be done through Ubuntu with hping3 and sqlmap installed for SYN flood and sql injection respectively.

### DoS attack-hping3:

The following line of command is launching a SYN flood denial of service attack to the target.

```
hping3 -c 15000 -d 120 -S -w 64 -p 80 --flood --rand-source <target ip address>
```

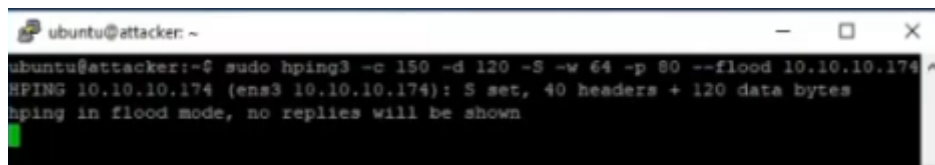


Figure 13. SYN flood

### SQL injection-sqlmap:

The following line of command is launching SQL injection through sqlmap to attack LAMP server:

```
sqlmap -u "http://10.10.10.174/phpmyadmin/admin/index.php?id=1; --level=1 --dbms=mysql --sql-
```



Figure 14. Sql injection

## Challenges

This section will describe details of challenges or problems we have encountered during the whole cloud setup process.

### Deploying OpenStack

Initially, we were thinking of deploying OpenStack on Google Cloud. This idea was appealing as we would get access to the recommended computing resources as per OpenStack documentation. OpenStack was therefore deployed on Google cloud, but we came across a lot of issues and decided to try other deployment methods outlined below. The following three methods were also attempted throughout the course of this project, and the installation of OpenStack was successful, but we encountered more other issues illustrated below.

- Ubuntu with OpenStack installer

Able to completely configure our network and instances but could not log into any of them. After several attempts, we moved on to use Packstack.

- Centos using Packstack

We have faced similar issues. After trimming down the modules we settled on this install command that did give us some additional success, but we encountered an issue with accessibility that will be further discussed (**platform issue**). We tried this install in different variations and host configurations so many times that we can script the install (Please view in appendix- **Installation Script with Packstack**)

```
packstack --allinone --default-password=inse6620S3cr3t --provision-demo=n  
--os-neutron-ovs-bridge-mappings=extnet:br-ex --os-neutron-ovs-bridge-  
interfaces=br-ex:eth0 --os-neutron-ml2-type-drivers=vxlan,flat
```

Figure 15. Packstack Command line Installation

- Ubuntu using TripleO/Ansible

Not a lot of time was put into this install. It is a framework that is meant to install an OpenStack lab on an OpenStack bare metal install. (OoO – OpenStack on OpenStack). There are some good tutorials on this, and we thought some carry-over might work because the virtual environment being installed in a virtual environment was similar to our attempts.

### **Solved Problem: Install scripts error**

Both the Packstack and OpenStack install scripts ran into issues, even when using the recommended software versions of the framework.

### **Solution examples:**

- A. Outfilter.py has an error that causes the installer to fail. We had to modify one of the write commands to permit the install to complete

- B. Wrapt and SimpleJson versions did not match the installer version and the installer failed to update them. We had to manually update both of them before running the installer in order for them to complete
- C. The Packstack install failed to properly configure the Horizon module and upon completion we had to manually modify the default Apache2 configuration file in order for Horizon to work properly

### **Unsolved Problem: No external access of Instances**

This was a major issue because it did not represent the environment we were trying to create and caused us not to be able to install or update any software. This occupied the bulk of our troubleshooting time as it was not a simple issue to resolve.

### **Attempt to solve:**

1. Pinpoint the packet occurs fails:

Our first test instance was very simple and occupied the tenant network. It had direct connectivity to the virtual router in OpenStack via the “private” network. Once we determined that the private network was configured properly and the subnet had the correct gateway matching the router interface, we proceeded to perform connectivity tests. We confirmed the instance could ping, itself (assuring the IP stack is OK), the private network gateway and the public network router interface. This last test failed.

2. Change rule to make entries allow ICMP and SSH

With a few Internet searches, we found out that OpenStack does not allow most traffic by default. The recommendation was to make entries in the default security groups to allow ICMP and SSH. We decided that we should open it completely to allow all testing. This allowed us to go a little further and we could now ping the host OpenStack server’s external bridge interface. We could not ping the provider network (a home network LAN) gateway.

### Manage Security Group Rules: default (ad9f8363-b75b-41eb-bf05-b18d96472244)

Displaying 8 items

<input type="checkbox"/> Direction	Ether Type	IP Protocol	Port Range	Remote IP Prefix	Remote Security Group
<input type="checkbox"/> Egress	IPv4	Any	Any	0.0.0.0/0	-
<input type="checkbox"/> Egress	IPv4	ICMP	Any	0.0.0.0/0	-
<input type="checkbox"/> Egress	IPv4	TCP	Any	0.0.0.0/0	-
<input type="checkbox"/> Egress	IPv6	Any	Any	::/0	-
<input type="checkbox"/> Ingress	IPv4	Any	Any	-	default
<input type="checkbox"/> Ingress	IPv4	ICMP	Any	0.0.0.0/0	-
<input type="checkbox"/> Ingress	IPv4	TCP	Any	0.0.0.0/0	-
<input type="checkbox"/> Ingress	IPv6	Any	Any	-	default

Displaying 8 items

Figure 16. Manage Security Group rules

3. New problem encounter

Using tcpdump, we determined that the traffic was never exiting the external bridge. Examining the host network leads to confusion. The external bridge, the internet facing interface and the home router were all on the same subnet, this led to the determination that if the host was set up properly, which



any documentation we found indicated, then it could not be a layer 3 issues and we would have to look at the bridging.

Figure below shows that the interface does not show an IP address because it is actually assigned to the external bridge and the internet facing interface is a member of the external bridge, which is the configuration recommended for provider type installations.

```

valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1460 qdisc mq master ovs-system state UP group default qlen 1000
   link/ether 42:01:0a:a2:01:02 brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1460 qdisc mq state UP group default qlen 1000
   link/ether 42:01:0a:a2:0a:02 brd ff:ff:ff:ff:ff:ff
   inet 10.162.10.2/32 brd 10.162.10.2 scope global dynamic eth1
       valid_lft 2698sec preferred_lft 2698sec
6: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
   link/ether 22:80:22:52:b9:df brd ff:ff:ff:ff:ff:ff
7: br-ex: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1460 qdisc noqueue state UNKNOWN group default qlen 1000
   link/ether 42:01:0a:a2:01:02 brd ff:ff:ff:ff:ff:ff
   inet 10.162.1.2/32 brd 10.162.1.2 scope global dynamic br-ex
       valid_lft 2172sec preferred_lft 2172sec
8: br-int: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
   link/ether 76:96:13:b5:c4:49 brd ff:ff:ff:ff:ff:ff
[root@inse6620 ~]#

```

Figure 17. No IP shown on Interface

### Unsolved Problem: OVS bridging

More internet investigation led to a number of possible issues revolving around bridging. The first revolved around port security. OpenStack restricts which traffic is permitted to traverse ports. Many OpenStack community members recommended disabling the port security, yet the method did not resolve the issue.

```

[root@openstack ~ (keystone_admin)]# openstack port list
+-----+-----+-----+-----+-----+
| ID | Name | MAC Address | Fixed IP Addresses | Status |
+-----+-----+-----+-----+-----+
| 33f8bcbf-84e1-4e70-a8af-ccff9bdab798 | | fa:16:3e:ef:9e:81 | ip_address='10.10.10.1', subnet_id='d3bbc387-675d-4784-ac31-257165f1549f' | ACTIVE |
| 62ff9a2c-e290-4c9e-bd66-37d93efc14c3 | | fa:16:3e:4f:6f:40 | ip_address='10.10.10.129', subnet_id='d3bbc387-675d-4784-ac31-257165f1549f' | DOWN |
| 710e1930-5456-4294-a9d2-6be0abf3eall | | fa:16:3e:61:82:61 | ip_address='192.168.2.202', subnet_id='b996497b-2aal-4839-b8cc-3dd0edbc6ef8' | N/A |
| 867a16c8-29a6-4f14-9f40-f830f989b0cb | | fa:16:3e:39:d3:a2 | ip_address='10.10.10.129', subnet_id='d3bbc387-675d-4784-ac31-257165f1549f' | ACTIVE |
| c8f0263-aacc-4dde-a99f-al667f0d022a | | fa:16:3e:9f:0f:3f | ip_address='172.16.1.20', subnet_id='62f63eb1-5dd3-4b71-a6d1-807bb8567fba' | ACTIVE |
| e237a6a3-f9fb-4cc0-8d40-771c2efc5a50 | | fa:16:3e:ef:40:f5 | ip_address='192.168.2.203', subnet_id='b996497b-2aal-4839-b8cc-3dd0edbc6ef8' | ACTIVE |
| f105b784-f40f-4031-b067-bd47640698e0 | | fa:16:3e:82:30:a1 | ip_address='10.10.10.2', subnet_id='d3bbc387-675d-4784-ac31-257165f1549f' | DOWN |
| fa335c66-a823-444f-bc66-a8c21f899c29 | | fa:16:3e:50:ca:b8 | ip_address='172.16.1.10', subnet_id='62f63eb1-5dd3-4b71-a6d1-807bb8567fba' | DOWN |
+-----+-----+-----+-----+-----+

[root@openstack ~ (keystone_admin)]# neutron port-update --no-security-groups 710e1930-5456-4294-a9d2-6be0abf3eall
neutron CLI is deprecated and will be removed in the future. Use openstack CLI instead.
Updated port: 710e1930-5456-4294-a9d2-6be0abf3eall
[root@openstack ~ (keystone_admin)]# neutron port-update 710e1930-5456-4294-a9d2-6be0abf3eall --port-security-enabled=False
neutron CLI is deprecated and will be removed in the future. Use openstack CLI instead.
Updated port: 710e1930-5456-4294-a9d2-6be0abf3eall
[root@openstack ~ (keystone_admin)]#

```

Figure 18. OVS bridging

Further investigation mentions that the internal and external bridge work using bridge mappings. One of the key mappings is an interconnect between the internal and external bridges called a patch peer. This virtual patch allows traffic on the internal bridge (all tenant-level instances) access to the internet via the external bridge. In the case of our installation of a Provider network installation of OpenStack, we believe the issues we are having revolve around this peer patch. Our installation appeared to be correct based on all of the documentation we have found but considering there is very little documentation for troubleshooting devstack installations on a virtual machine, we concluded that a different approach would be a better idea.

### Platform issue: Google cloud

Google only provides one routable IP address which we need several to build a cloud. The reason why OpenStack can be installed is that a virtual bridge was set during the installation which has access to the internet via a tunnel from the bridge interface to the internet-connected interface. It is possible to obtain more IP address from Google; however, they are aliases on a different subnet, and it does not work out of the box.

## Building up virtual bridge

The way open stack allowed instances to communicate is that it creates a virtual bridge that all devices connect to and it then provides security to ensure only the correct devices can speak with each other. Our virtual bridge is not working because the MTU size of the packet is too large for the bridge. When following instruction on document attempting to fix the issues, we lost entire interface setup. The interface `enp0s8` is the interface I am using to connect to the Internet. As below, it has no IP address, but `br-ex` (external bridge) that it has the same MAC address as `enp0s8` and is assigned an IP. Normally this would be very strange but that is how the virtualization works.

```
stack@openstack:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:d1:d2:c1 brd ff:ff:ff:ff:ff:ff
    inet 192.168.56.102/24 brd 192.168.56.255 scope global dynamic enp0s3
        valid_lft 516sec preferred_lft 516sec
    inet6 fe80::a00:27ff:fed1:d2c1/64 scope link
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel master ovs-system state UP
    link/ether 08:00:27:8c:c6:f0 brd ff:ff:ff:ff:ff:ff
    inet6 fe80::a00:27ff:fe8c:c6f0/64 scope link
        valid_lft forever preferred_lft forever
4: virbr0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default qlen 1000
    link/ether 52:54:00:12:2e:0f brd ff:ff:ff:ff:ff:ff
    inet 192.168.122.1/24 brd 192.168.122.255 scope global virbr0
        valid_lft forever preferred_lft forever
5: virbr0-nic: <BROADCAST,MULTICAST> mtu 1500 qdisc fq_codel master virbr0 state DOWN group default qlen 1000
    link/ether 52:54:00:12:2e:0f brd ff:ff:ff:ff:ff:ff
6: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 1a:1c:74:94:aa:95 brd ff:ff:ff:ff:ff:ff
7: br-int: <BROADCAST,MULTICAST> mtu 1450 qdisc noop state DOWN group default qlen 1000
    link/ether 7a:b0:13:2c:56:42 brd ff:ff:ff:ff:ff:ff
8: br-ex: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN group default qlen 1000
    link/ether 08:00:27:8c:c6:f0 brd ff:ff:ff:ff:ff:ff
    inet 192.168.2.88/24 brd 192.168.2.255 scope global br-ex
        valid_lft forever preferred_lft forever
    inet 172.24.4.1/24 scope global br-ex
        valid_lft forever preferred_lft forever
    inet6 2001:db8::2/64 scope global
        valid_lft forever preferred_lft forever
    inet6 fe80::9034:a9ff:fea4:cd4e/64 scope link
        valid_lft forever preferred_lft forever
```

Figure 19. OVS bridging

Below figure shows the virtual connection between the internal bridge and the external bridge is in place which meets what document indicates is correct; however, there is no internet access.

```
stack@openstack:~$ sudo tail -f /var/log/kern.log
Jul 25 16:11:34 openstack kernel: [83383.420472] br-int: dropped over-mtu packet: 1460 > 1450
Jul 25 16:11:34 openstack kernel: [83383.423470] br-int: dropped over-mtu packet: 1456 > 1450
Jul 25 16:11:38 openstack kernel: [83387.420330] br-int: dropped over-mtu packet: 1465 > 1450
Jul 25 16:11:38 openstack kernel: [83387.422853] br-int: dropped over-mtu packet: 1475 > 1450
Jul 25 16:11:54 openstack kernel: [83403.418913] br-int: dropped over-mtu packet: 1465 > 1450
Jul 25 16:11:54 openstack kernel: [83403.421405] br-int: dropped over-mtu packet: 1475 > 1450
Jul 25 16:12:26 openstack kernel: [83435.415658] br-int: dropped over-mtu packet: 1465 > 1450
Jul 25 16:12:26 openstack kernel: [83435.418038] br-int: dropped over-mtu packet: 1475 > 1450
Jul 25 16:13:30 openstack kernel: [83499.409015] br-int: dropped over-mtu packet: 1465 > 1450
Jul 25 16:13:30 openstack kernel: [83499.411387] br-int: dropped over-mtu packet: 1475 > 1450
```

Figure 20. Virtual connection list



## Summary

In conclusion, the team was able to accomplish what was presented in the initial proposal. The project includes an OpenStack virtual machine running multiples instances such as Snort, Splunk, LAMP server and an attack machine. With that, we were able to demonstrate the different instance worked together as the attacks perform on the target were detected by Snort and showed in Splunk logs.

Each tool required a research phase to understand how it works and how it can be integrated to OpenStack.

During the project, many challenges arose especially in the OpenStack installation process, but we were able to find workarounds and solutions that bring us in a place where all the tools could be integrated. To accomplish that, the team members were able to divide the tasks amongst themselves to work in parallel and be able to add their part when the OpenStack instance was ready to include a new tool in its environment.

Finally, while time constraints prevent us to further investigate several issues that we came across throughout the course of this project, we are however in a position to deliver a project of quality and have successfully completed the project as per the initial project proposal inline with the project requirement.

## ***Appendix***

### **Contribution table(tasks)**

<b>Team member</b>	<b>Task</b>
Daniel Kabagema	OpenStack, research on integration of the tools and presentation
James Kearney	OpenStack, integration of the tools and presentation
Ho-Yang Chang	Splunk installation, documentation
Omer Mujtaba	LAMP, Attack nodes, Splunk configuration, Presentation
Pooja Singh	Snort_Document
Sami Islam	Sql injection_ Documentation
Simon Benoit	Documentation format, LAMP, Attack nodes, team management, Snort
Saeid Jamshidi	Snort

## Installation Script with Packstack

```
#!/bin/bash

#####
#create backup of eth0 to restore after a bad change and set dirty bit for bad settings
#if the dirty bit exists at startup the backup config is restored
#####
mkdir -p /var/backup && cp -p /etc/sysconfig/network-scripts/ifcfg-eth0 /var/backup/
touch /var/backup/badcfg

#####
# create boot script
#####
echo 'if test -f "/var/backup/badcfg" ; then' > badStartTest.sh
echo "    cp -p /var/backup/ifcfg-eth0 /etc/sysconfig/network-scripts/" >> badStartTest.sh
echo "    systemctl restart network" >> badStartTest.sh
echo "else" >> badStartTest.sh
echo "    echo ***** Clean restart *****" >> badStartTest.sh
echo fi >> badStartTest.sh

chmod 755 badStartTest.sh
cp -p badStartTest.sh /var/backup/
#####

#####
# Disable IPV6
#####
echo "net.ipv6.conf.all.disable_ipv6=1" >> /etc/sysctl.conf
echo "net.ipv6.conf.default.disable_ipv6=1" >> /etc/sysctl.conf
sed -i 's/#AddressFamily any/AddressFamily inet/g' /etc/ssh/sshd_config
sysctl -p

#####
# make SSH keys for root and allow root login
#####
cd ~

cat /root/.ssh/id_rsa.pub > /root/.ssh/authorized_keys
cat /root/.ssh/id_rsa
sed -i 's/PermitRootLogin no/PermitRootLogin yes/g' /etc/ssh/sshd_config

if grep -q "PermitRootLogin yes" /etc/ssh/sshd_config; then
    systemctl restart sshd
else
    echo "Modification of SSHD failed" | tee install.log
    exit 1
fi

echo *****
echo *****
echo ***** COPY THE SSH PRIVATE KEY NOW *****
echo ***** Press any key once it is copied *****
echo *****
echo *****
echo *****
read -n 1 -s #wait for keypress
```

```
#####

#####
# Download necessary packages
#####
yum install cloud-init -y
yum install centos-release-OpenStack-train -y
yum install OpenStack-packstack -y
yum update -y
yum upgrade
yum downgrade leatherman -y
#####

#####
#disable problematic services and turn on networking
#####
systemctl disable NetworkManager
systemctl stop NetworkManager
systemctl disable firewalld.service
systemctl stop firewalld.service
systemctl enable network
systemctl start network
#####

#####
#disable selinux
#####
setenforce 0 # turn it off now

#keep it turned off after reboot
sed -i 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/selinux/config

if grep -q "SELINUX=disabled" /etc/selinux/config; then
    echo "Selinux disabled permanently" | tee install.log
else
    echo "Selinux NOT disabled permanently -- investigate" | tee install.log
fi
#####

#####
#Install OpenStack --- br-ex:eth0 may be a poential problem
#####
packstack --allinone --default-password inse6620S3cr3t --provision-demo=n --os-neutron-ovs-
bridge-mappings=extnet:br-ex --os-neutron-ovs-bridge-interfaces=br-ex:eth0 --os-neutron-ml2-
type-drivers=vxlan,flat

mv /etc/httpd/conf.d/15-default.conf /etc/httpd/conf.d/15-default.old
systemctl restart httpd

cat /root/.ssh/id_rsa
echo "Remember to create @reboot cron job"
echo "Remember to delete the file /var/backup/badcfg if everything is working"
```

## Installation Script of Snort

1. *Install Prerequisites for Snort*  
`sudo apt install -y gcc libpcap-dev zlib1g-dev liblua5.1-dev \`  
`libpcap-dev openssl libssl-dev libnghttp2-dev libdumbnet-dev \`  
`bison flex libdnet autoconf libtool`
2. *Making a temporary download folder*  
`mkdir ~/snort_src && cd ~/snort_src`
3. *Download the latest Data Acquisition library (DAQ) source package from the Snort website with the wget command underneath. Snort itself uses DAQ to make abstract calls to packet capture libraries.*  
`wget https://www.snort.org/downloads/snort/daq-2.0.7.tar.gz`
4. *When completed, the source code can be extracted and jump into the new directory with the following commands.*  
`tar -xvzf daq-2.0.7.tar.gz`  
`cd daq-2.0.7`
5. *Configure the installation with sourcefire enabled, run make and make install.*  
`./configure --enable-sourcefire && make && sudo make install`

## Snort rules set for our cloud

```
alert tcp any any -> any $HTTP_PORTS (msg:"Modified regex for detection of SQL meta-
characters";flow:to_server,established;pcre:"/((\%3D)|(|=))[\n]*(\%27)|(|'|)|(|-)|(|%3B)|(|;))/i";
classtype:Web-application-attack; sid:910000;rev:5;)
```

```
alert tcp any any -> any $HTTP_PORTS (msg:"Regex for typical SQL Injection
attack";flow:to_server,established;pcre:"\w*(\%27)|(|'|)|(|%6F)|o|(|%4F)|(|%72)|r|(|%52))/ix"
; classtype:Web-application-attack; sid:910001;rev:5;)
```

```
alert tcp any any -> any $HTTP_PORTS (msg:"Regex for detecting SQL Injection with the
UNION keyword";flow:to_server,established;pcre:"/((\%27)|(|'|))union/ix"; classtype:Web-
application-attack; sid:910002;rev:5;)
```

```
alert tcp any any -> any $HTTP_PORTS (msg:"Regex for detecting SQL Injection attacks on
a MS SQL Server";flow:to_server,established;pcre:"/exec(\s|+)+(s|x)p\w+/ix";
classtype:Web-application-attack; sid:910003;rev:5;)
```

```
alert tcp any any -> any $HTTP_PORTS (msg:"Regex for simple CSS
attack";flow:to_server,established;pcre:"/((\%3C)|<)(\%2F)|V)*[a-z0-9\%]+((\%3E)|>)/ix";
classtype:Web-application-attack; sid:910004;rev:5;)
```

```
alert tcp any any -> any $HTTP_PORTS (msg:"Regex for img src CSS
attack";flow:to_server,established;pcre:"/((\%3C)|<)(\%69)|i|(|%49)|(|%6D)|m|(|%4D)|(|%6
7)|g|(|%47))[\n]+((\%3E)|>)/l"; classtype:Web-application-attack; sid:910005;rev:5;)
```

```
alert tcp any any -> any $HTTP_PORTS (msg:"Paranoid regex for CSS
attacks";flow:to_server,established;pcre:"/((\%3C)|<)[\n]+((\%3E)|>)/l"; classtype:Web-
application-attack; sid:910006;rev:5;)
```

## References

- [1] OpenStack, "Open vSwitch: Provider networks" OpenStack.  
[Website]. Available: <https://docs.OpenStack.org/ocata/networking-guide/deploy-ovs-provider.html>, Accessed on: August 9, 2020.
- [2] OpenStack, "Using DevStack with neutron Networking" OpenStack.  
[Website]. Available: <https://docs.OpenStack.org/devstack/ocata/guides/neutron.html>, Accessed on: August 9, 2020.
- [3] Ubuntu, "Single-node OpenStack deployment" Ubuntu.  
[Website]. Available: <https://ubuntu.com/OpenStack/install#single-node-deployment>, Accessed on: August 9, 2020.
- [4] J. Orti Alcaine, "Run a OpenStack all-in-one in Google Cloud" Apuntes de root.  
[Blog]. Available: <https://apuntesderootblog.wordpress.com/2018/03/20/run-a-OpenStack-all-in-one-in-google-cloud/>, Accessed on: August 9, 2020.
- [5] J. Langemak, "Building an OpenStack home lab – Installing OpenStack" Das Blinken Lichten - Opentstack.  
[Blog]. Available: <https://www.dasblinkenlichten.com/category/OpenStack/>, Accessed on: August 9, 2020.
- [6] David Field, "Installing OpenStack Devstack" David Field blog.  
[Blog]. Available: <https://tech.davidfield.co.uk/installing-OpenStack/>, Accessed on: August 9, 2020.
- [7] David Mahler, "OpenStack" Youtube Channel-David Mahler.  
[Website]. Available: <https://docs.OpenStack.org/ocata/networking-guide/deploy-ovs-provider.html>, Accessed on: August 9, 2020.
- [8] TheSkillPedia, "OpenStack tutorial Lab 1 Machines Setup with Networking | #OpenStack Rocky on Ubuntu" TheSkillPedia.  
[Website]. Available: <https://www.youtube.com/watch?v=AQ8igcN2p18>, Accessed on: August 9, 2020.
- [9] N. Dietrich, "Snort 3.0.1 on Ubuntu 18 & 20" (2020) [Online]. Available: [https://snort-org-site.s3.amazonaws.com/production/document\\_files/files/000/000/251/original/Snort\\_3\\_on\\_Ubuntu.pdf?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIAXACIED2SPMSC7GA%2F20200809%2Fus-east-1%2Fs3%2Faws4\\_request&X-Amz-Date=20200809T195953Z&X-Amz-Expires=172800&X-Amz-SignedHeaders=host&X-Amz-Signature=354623d800b6bc961c57381ecc7647172d0dd2eb4867d394ab0e75b99e8798dc](https://snort-org-site.s3.amazonaws.com/production/document_files/files/000/000/251/original/Snort_3_on_Ubuntu.pdf?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIAXACIED2SPMSC7GA%2F20200809%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20200809T195953Z&X-Amz-Expires=172800&X-Amz-SignedHeaders=host&X-Amz-Signature=354623d800b6bc961c57381ecc7647172d0dd2eb4867d394ab0e75b99e8798dc), Accessed on: August 9, 2020.
- [10] Stack Exchange, "Snort rule for syn flood attacks - Limiting number of alerts" Stack Exchange Information Security.  
[Website]. Available: <https://security.stackexchange.com/questions/204936/snort-rule-for-syn-flood-attacks-limiting-number-of-alerts>, Accessed on: August 9, 2020.
- [11] J. Ruostemaa, "How to install Snort on Ubuntu" Upcloud.  
[Website]. Available: <https://upcloud.com/community/tutorials/install-snort-ubuntu/>, Accessed on: August 9, 2020.
- [12] SplunkBase, "OpenStack App for Splunk" SplunkBase.  
[Website]. Available: <https://splunkbase.splunk.com/app/2904/>, Accessed on: August 9, 2020.
- [13] Splunk, "Splunk Security Investigations, Part 1: Threat Detection" Splunk.  
[Website]. Available: [https://www.youtube.com/watch?v=7DRHt8LJN\\_g](https://www.youtube.com/watch?v=7DRHt8LJN_g), Accessed on: August 9, 2020.
- [14] Splunk, "Splunk documentation" Splunk.  
[Website]. Available: <https://docs.splunk.com/Documentation>, Accessed on: August 9, 2020.
- [15] Splunk, "Splunk Fundamentals 1" Splunk.  
[Website]. Available: [https://www.splunk.com/en\\_us/training/free-courses/splunk-fundamentals-1.html](https://www.splunk.com/en_us/training/free-courses/splunk-fundamentals-1.html), Accessed on: August 9, 2020.

- [15] M. Drake, "How To Install Linux, Apache, MySQL, PHP (LAMP) stack on Ubuntu 18.04" Digital Ocean community.  
[Website]. Available: <https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-ubuntu-18-04>, Accessed on: August 9, 2020.
- [15] Firewall, "How to perform tcp syn flood dos attack & detect it with wireshark - kali linux hping3" Firewall. [Website]. Available: <http://www.firewall.cx/general-topics-reviews/network-protocol-analyzers/1224-performing-tcp-syn-flood-attack-and-detecting-it-with-wireshark.html>, Accessed on: August 9, 2020.
- [16] Author Janne Ruostemaa, "How to install Snort on Ubuntu",  
[Website]. Available: <https://upcloud.com/community/tutorials/install-snort-ubuntu/>