National College of Ireland

Configuration Manual For SDN Based DDOS Attack Simulation Platform

MSc Research Project
Cloud Computing

Vishal Kumar Singh
Student ID: x18201687

School of Computing
National College of Ireland

Supervisor: Vikas Sahni

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Vishal Kumar Singh
Student ID:	x18201687
Programme:	Cloud Computing
Year:	2020
Module:	MSc Research Project
Supervisor:	Vikas Sahni
Submission Due Date:	17/09/2020
Project Title:	Configuration Manual For SDN Based DDOS Attack Simulation Platform
Word Count:	1667
Page Count:	12

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

I agree to an electronic copy of my thesis being made publicly available on TRAP the National College of Ireland's Institutional Repository for consultation.

Signature:	
Date:	16th August 2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual For SDN Based DDOS Attack Simulation Platform

Vishal Kumar Singh
x18201687

1 Introduction

Software defined networks has various protocols and controllers, each of those are designed to perform in a certain way and provide efficiency and flexibility in a particular aspect. The presented method is implemented using the most popular and out performing tools for the detection and mitigation of DDOS attacks in a software defined network.
Openflow Protocol is the most popular and standard protocol for software defined networks, hence openVswitch is used for this project.As the presented method is a combination of statistical and machine learning methods, the logic and techniques are programmed using python. Statistical method include parameters such as speed of source IP, speed of flow entires and ratio of flowpair entries, all of these logic is programmed in the controller.
Ryu Controller is an open-source python based programmable controller, which is used to define the rules and logic for the switches to follow in the methodology.
Mininet is a network simulator and creates a virtual network topology with controller, switches and hosts, in this work a single openVswitch with 10 and 25 hosts are created for multiple tests.
Hping3 is a packet generator which generates TCP/IP traffic in the network, it is mostly used to test network security. Normal and attack traffic scripts are written to generate traffic automatically using this tool.
Iperf is also a network traffic generator and network performance tester, which in this work is used to generate traffic manually.

2 Simulation Platform Setup

This Section involves the steps and procedure to install all the packages and software required to implement the project. The Platform setup is done on Ubuntu 20.04.1 LTS operating system.
The project demonstration and simulation was using the following tools:-

- Openflow Protocol For SDN
- Ryu Controller
- Mininet
- Hping3

- Iperf

Before moving forward with packages installation make sure you have the updated ubuntu OS and linux libraries with pyhton 3.8 and python 2.7 installed. Open terminal and type in the following commands:-

-sudo apt-get update

-sudo apt-get upgrade

-sudo apt install python2

-sudo apt install python3

Openflow protocol For SDN or OpenVswitch has to be installed as it is the standard protocol for software defined network- Switch (2009).[1]
Open Terminal and type in the following command:-

-sudo apt-get install openvswitch-switch

Give yes(Y) where ever it is asked, and to check the version and confirm the installation type in:-

-ovs-vsctl –version

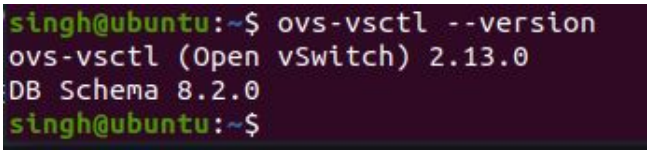

Figure 1: OpenVswitch Version.

Ryu Controller has to be installed to see the process of detection and mitigation stages. To install ryu controller you need to install PIP of python to install python packages, as ryu is a python based controller it has to be installed using PIP. To install PIP and Ryu controller type in these commands in terminal.[2]

-sudo apt install python3-pip

-pip3 –version

-sudo pip3 install ryu

-ryu-manager –version

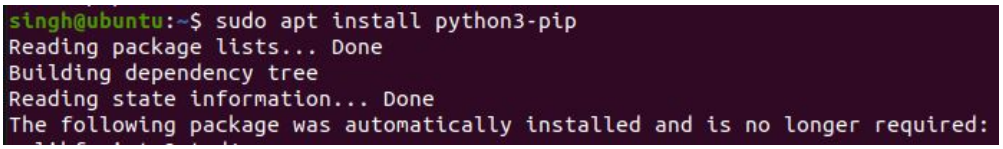

Figure 2: Pip Installation

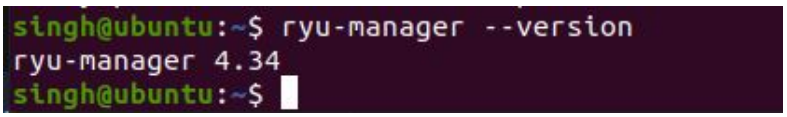

Figure 5: Ryu Check.

[1]Openflow switch: `https://www.opennetworking.org/tag/openflow/`
[2]Ryu Controller: `https://ryu-sdn.org/`

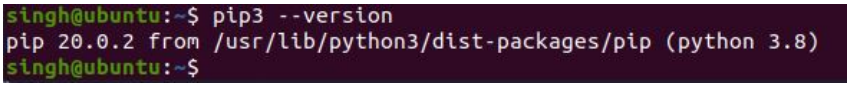

Figure 3: Pip Check

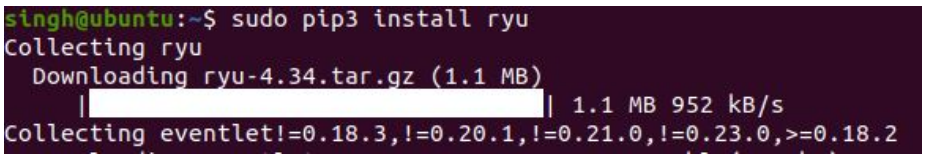

Figure 4: Ryu Installation.

Mininet is a network simulator and creates virtual network topology for software defined networks.[3]

-sudo apt-get install mininet

-mn –version

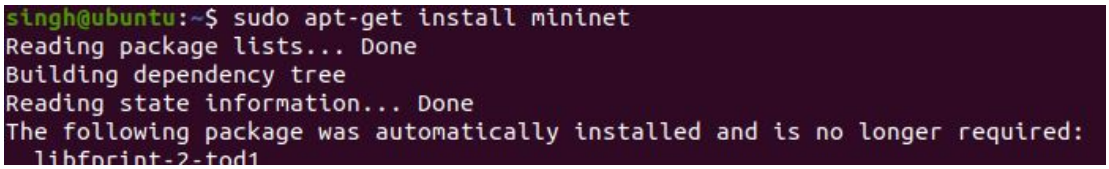

Figure 6: Mininet Installation.

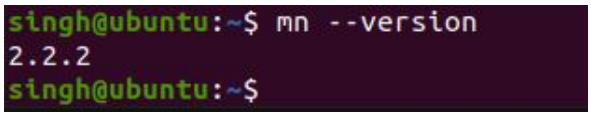

Figure 7: Mininet Check.

Hping3 is a network packet generator and traffic generator for TCP/IP protocol, mostly used for network testing[4]. **Iperf** is network traffic performance tool to generate traffic and monitor it.[5]

-sudo apt-get install iperf

-sudo apt-get install hping3

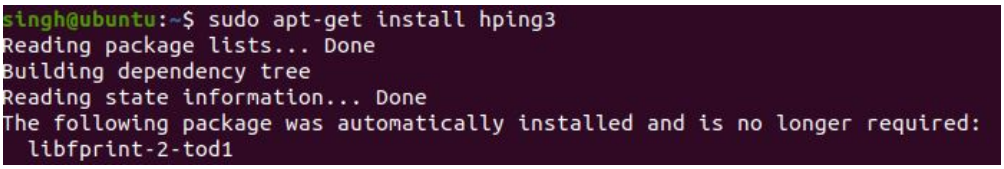

Figure 9: Hping3 Installation.

Ryu packages has to be installed using pip as the ryu controller needs Python packages to run in this environment.

[3]Mininet Simulator: `http://mininet.org/`

[4]Hping3 Tool: `https://tools.kali.org/information-gathering/hping3`

[5]Iperf Tool: `https://iperf.fr/`

```
singh@ubuntu:~$ sudo apt-get install iperf
Reading package lists... Done
Building dependency tree
Reading state information... Done
iperf is already the newest version (2.0.13+dfsg1-1build1).
iperf set to manually installed.
The following package was automatically installed and is no longer required:
  libfprint-2-tod1
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
singh@ubuntu:~$ iperf --version
iperf version 2.0.13 (21 Jan 2019) pthreads
singh@ubuntu:~$
```

Figure 8: Iperf Installation.

-sudo pip3 install numpy
-sudo pip3 install sklearn

```
singh@ubuntu:~/sdn$ sudo pip3 install numpy
Collecting numpy
  Downloading numpy-1.19.1-cp38-cp38-manylinux2010_x86_64.whl (14.5 MB)
     |████████████████████████████████| 14.5 MB 5.1 MB/s
Installing collected packages: numpy
Successfully installed numpy-1.19.1
singh@ubuntu:~/sdn$
```

Figure 10: Numpy Package Installation.

```
singh@ubuntu:~/sdn$ sudo pip3 install sklearn
Collecting sklearn
  Downloading sklearn-0.0.tar.gz (1.1 kB)
Collecting scikit-learn
  Downloading scikit_learn-0.23.2-cp38-cp38-manylinux1_x86_64.whl (6.8 MB)
     |████████████████████████████████| 6.8 MB 2.8 MB/s
Collecting threadpoolctl>=2.0.0
```

Figure 11: Sklearn Package Installation.

These are all the tools and packages required for the project to run and simulation to work. In the next sections we will see how to run attacks and obtain results.

3 Traffic Data Collection

Traffic data has to collected and stored in a file for the SVM and decision tree machine learning algorithms to analyse and predict the attack traffic.

3.1 Normal Traffic Data

Open the controller.py file and change the APP_TYPE=0 and TEST_TYPE=0(**See the line numbers from the image below**), here 0 is for data collection and 1 is for attack detection in app_type and test_type 0 is for normal traffic data collection and 1 is for

attack traffic data collection. Interval is the data collection intervals in seconds which is set to 3 seconds.

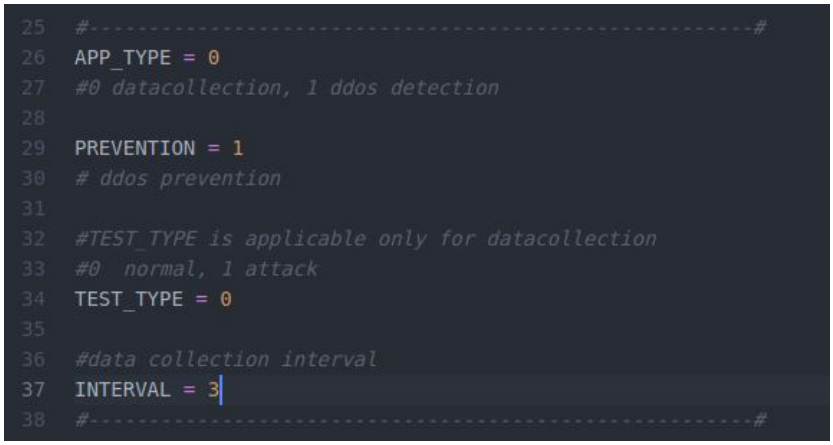

Figure 12: Controller.py Code.

Open the topo.py file and change the TEST_TYPE=normal, to generate normal traffic in the network (Test type "manual" is to generate traffic manually in mininet using Iperf tool) and TEST_TIME=600, to collect data for 600 seconds of duration-minimum 400 seconds is recommended for better accuracy. **Use and run file topo25.py to have 25 hosts network topoloy for the tests.**

Figure 13: Topo.py Code.

Open Terminal window and make sure you are in the project source codes folder and then start the ryu manager for traffic collection.
-ryu-manager controller.py

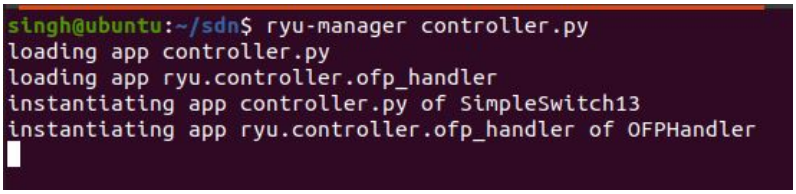

Figure 14: Start Ryu Controller.

Open another terminal window and start the topology for normal traffic generation.
-sudo python2 topo.py

```
singh@ubuntu:~/sdn$ sudo python topo.py
Connecting to remote controller at 127.0.0.1:6653
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Adding switches:
s1
*** Adding links:
(5.00Mbit) (5.00Mbit) (h1, s1) (5.00Mbit) (5.00Mbit) (h2, s1) (5.00Mbit) (5.00Mbit) (h3, s1) (5.00Mbit) (5.00Mbit) (
6, s1) (5.00Mbit) (5.00Mbit) (h7, s1) (5.00Mbit) (5.00Mbit) (h8, s1) (5.00Mbit) (5.00Mbit) (h9, s1) (5.00Mbit) (5.00
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Starting controller
c1
*** Starting 1 switches
s1 ...(5.00Mbit) (5.00Mbit) (5.00Mbit) (5.00Mbit) (5.00Mbit) (5.00Mbit) (5.00Mbit) (5.00Mbit) (5.00Mbit)
Generating NORMAL Traffic.......
```

Figure 15: Normal Traffic Generating.

Below image shows the normal traffic is being generated and stored a CSV file with flow counts and different features parameters count in brackets.

```
singh@ubuntu:~/sdn$ ryu-manager controller.py
loading app controller.py
loading app ryu.controller.ofp_handler
instantiating app controller.py of SimpleSwitch13
instantiating app ryu.controller.ofp_handler of OFPHandler
['08/13/2020, 05:27:00', '7', '4', '1.0']
['08/13/2020, 05:27:03', '4', '2', '1.0']
['08/13/2020, 05:27:06', '8', '2', '1.0']
['08/13/2020, 05:27:09', '2', '0', '1.0']
['08/13/2020, 05:27:12', '2', '1', '1.0']
['08/13/2020, 05:27:15', '10', '0', '1.0']
['08/13/2020, 05:27:18', '4', '0', '1.0']
['08/13/2020, 05:27:21', '0', '0', '1.0']
```

Figure 16: Controller receiving traffic.

3.2 Attack Traffic Data

Open the controller.py file and change the APP_TYPE=0 and TEST_TYPE=1(**See the line numbers from the image below**), here 0 is for data collection and 1 is for attack detection in app_type and test_type 0 is for normal traffic data collection and 1 is for attack traffic data collection. Interval is the data collection intervals in seconds which is set to 3 seconds.

```
26  APP_TYPE = 0
27  #0 datacollection, 1 ddos detection
28
29  PREVENTION = 1
30  # ddos prevention
31
32  #TEST_TYPE is applicable only for datacollection
33  #0  normal, 1 attack
34  TEST_TYPE = 1
35
36  #data collection interval
37  INTERVAL = 3
38  #-----------------------------------------------------#
```

Figure 17: Controller.py Code.

Open the topo.py file and change the TEST_TYPE=attack, to generate attack traffic in the network and TEST_TIME=600, to collect data for 600 seconds of duration-minimum 400 seconds is recommended for better accuracy.

Figure 18: Topo.py Code.

Open Terminal window and make sure you are in the project source codes folder and then start the ryu manager for traffic collection.
-ryu-manager controller.py

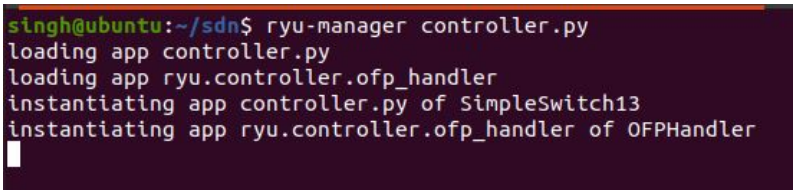

Figure 19: Start Ryu Controller.

Open another terminal window and clean the previous traffic generated history of mininet as this traffic is different from normal traffic, we need to clean the mininet history everytime we change the traffic generator type, and then start the topology for attack traffic generation.
Clean mininet history:- -sudo mn -c

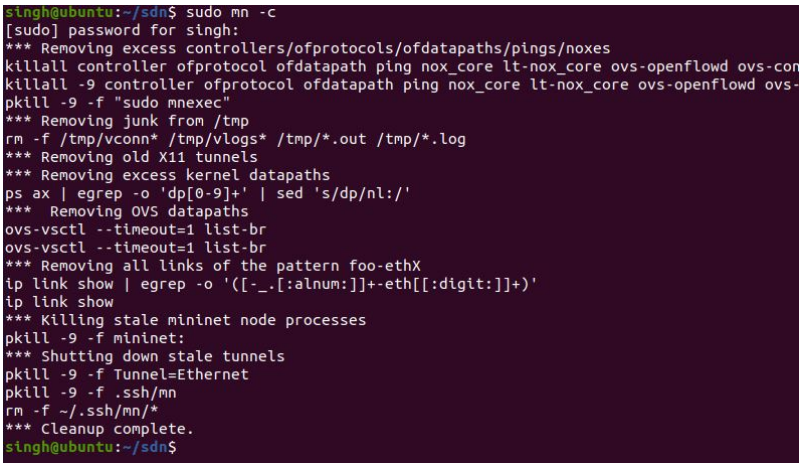

Figure 20: Clean Mininet History.

-sudo python2 topo.py

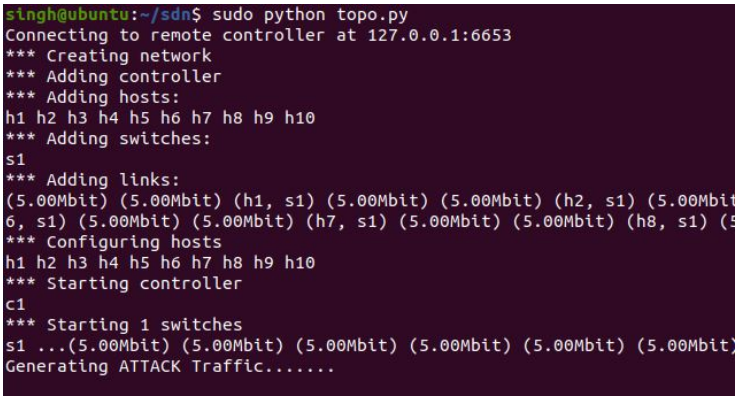

Figure 21: Attack Traffic Generating.

Below image shows the attack traffic is being generated and stored a CSV file with high flow counts and different features parameters count in brackets.

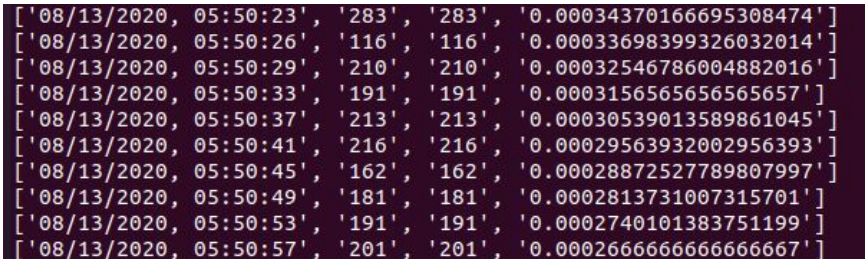

Figure 22: Controller receiving traffic.

4 DDOS Attack Detection and Mitigation Simulation

This section we will see the steps involved in simulating DDOS attack and see how the controller detects the attack traffic and mitigates it.

4.1 Normal Traffic Detection

Open controller.py file and only change the APP_TYPE=1, which is for attack detection.

Figure 23: Controller.py Code.

Open Topo.py file and chenge the TEST_TYPE=normal for generating normal traffic and keep the TEST_TIME=300, minimum of 3 seconds is enough for the controller to detect as the intervals are kept as 3 seconds.

```
17
18  TEST_TIME = 300 #seconds
19  TEST_TYPE = "normal"
20  #normal, attack, manual
```

Figure 24: Topo.py Code.

Open Terminal window and make sure you are in the project source codes folder and then start the ryu manager for traffic collection.
-ryu-manager controller.py

```
singh@ubuntu:~/sdn$ ryu-manager controller.py
loading app controller.py
loading app ryu.controller.ofp_handler
instantiating app controller.py of SimpleSwitch13
instantiating app ryu.controller.ofp_handler of OFPHandler
```

Figure 25: Start Ryu Controller.

Open another terminal window and start the topology to generate normal traffic.
-sudo python2 topo.py

```
singh@ubuntu:~/sdn$ sudo python topo.py
Connecting to remote controller at 127.0.0.1:6653
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Adding switches:
s1
*** Adding links:
(5.00Mbit) (5.00Mbit) (h1, s1) (5.00Mbit) (5.00Mbit) (h2, s1) (5.00Mbit) (5.00Mbit) (h3, s1) (5.00Mbit) (5.00Mbit) (
6, s1) (5.00Mbit) (5.00Mbit) (h7, s1) (5.00Mbit) (5.00Mbit) (h8, s1) (5.00Mbit) (5.00Mbit) (h9, s1) (5.00Mbit) (5.00
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8 h9 h10
*** Starting controller
c1
*** Starting 1 switches
s1 ...(5.00Mbit) (5.00Mbit) (5.00Mbit) (5.00Mbit) (5.00Mbit) (5.00Mbit) (5.00Mbit) (5.00Mbit) (5.00Mbit)
Generating NORMAL Traffic.......
```

Figure 26: Normal Traffic Generating.

From the below image it is shown that the controller and SVM is predicting it as normal traffic.

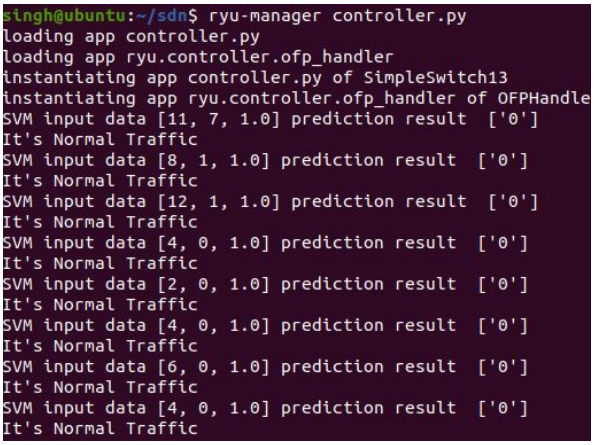

Figure 27: Normal Traffic Detection.

4.2 Attack Traffic Detection and Mitigation

No change in controller.py as we are still detecting for attacks.

Open Topo.py file and chenge the TEST_TYPE=attack for generating attack traffic and keep the TEST_TIME=300, minimum of 3 seconds is enough for the controller to detect as the intervals are kept as 3 seconds.

Figure 28: Topo.py Code.

Open Terminal window and make sure you are in the project source codes folder and then start the ryu manager for traffic collection.
-ryu-manager controller.py

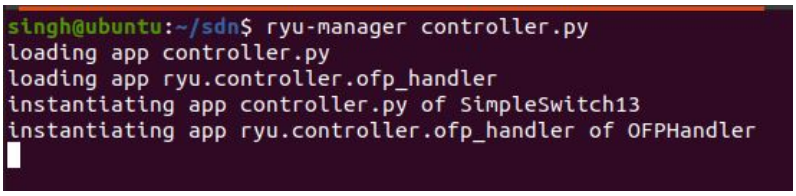

Figure 29: Start Ryu Controller.

Open another terminal window and start the topology to generate attack traffic.
-sudo python2 topo.py

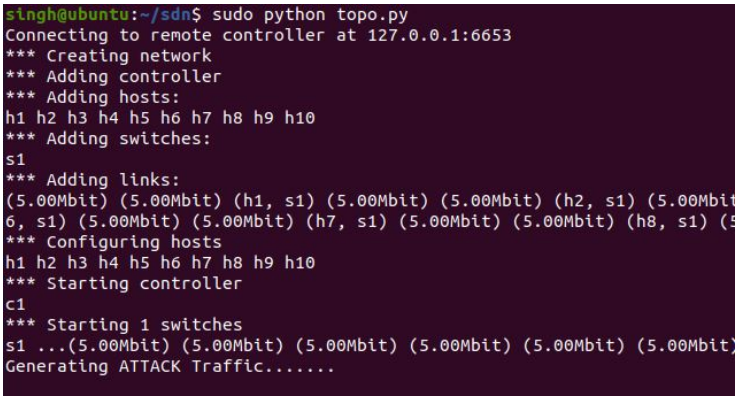

Figure 30: Attack Traffic Generating.

From the below image it is shown that the controller and SVM is predicting it as attack traffic and blocking the port from which attack is incoming. Then the rest traffic is predicted as normal traffic.

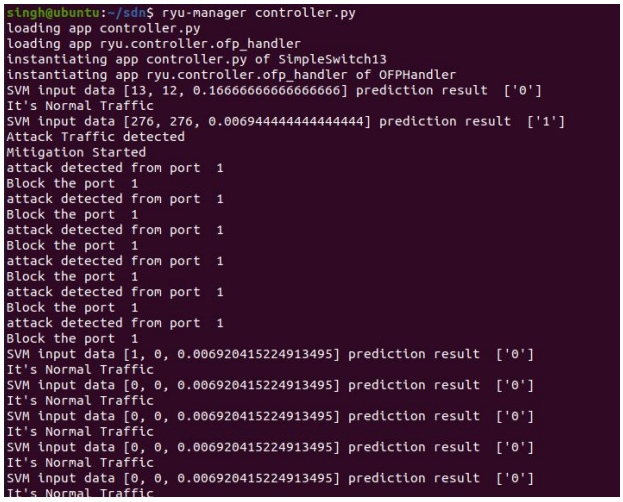

Figure 31: DDOS Attack Traffic Detection and Mitigation.

The controller is programmed to block the port for 120 seconds and then again the port which is blocked is unblocked. But if the attack traffic is still incoming then it again blocks the port. Shown in image below.

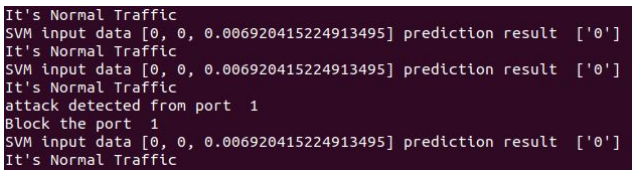

Figure 32: Blocking Port Again.

5 Accuracy and Detection rate calculation

I have attached a separate python files for checking the accuracy of the DDOS attack detection and the malicious traffic detection rate, and a python script to generate graphs of SVM machine learning algorithm prediction.

Make sure you are in the analysis folder and you have copied the results.CSV file which is generated from the previous steps and copy it in this analysis folder. Run the accuracy_score.py file. The accuracy achieved from this proposed method is shown in the image. It also shows the cross validation score.
-python accuracy_score.py

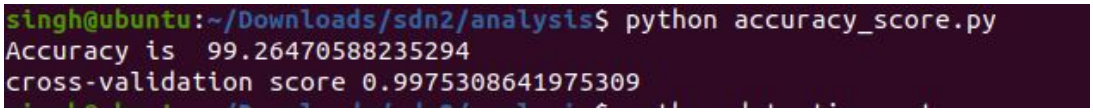

Figure 33: Accuracy Score.

To calculate the detection rate of this method, in the same folder run the detection_rate.py. -python detection_rate.py

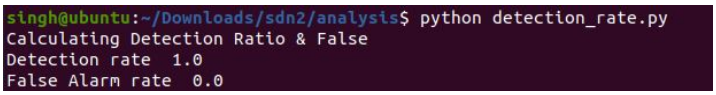

Figure 34: Detection Ratio.

To generate graph just run the graph.py in the same folder and the graphs will be generated and saved in the folder.

6 Decision Tree Algorithm Changes

As python has inbuilt libraries for both Support vector machine (SVM) and Decision tree machine learning algorithms, only the SVM.py file in the project source code files has to be changed. All the other requirements are already programmed in the controller.
Open SVM.py and do the following changes:-
Remove the comment from line 20 and add comment to line 18, and no changes to anymore files for detection and mitigation. Follow the steps from section 2 again for prediction results from Decision tree ML algorithm.

Figure 35: Decision Tree ML Algorithm change.

References

Switch, O. (2009). Version 1.0. 0 (wire protocol 0x01), *Open Networking Foundation* .