**Report on**

**DDoS Detection Using Entropy Computation**

***By***

***SUDHANSHU RANJAN***

***3rd Year B.tech CSE***

***(NIT SRINAGAR)***

***Submitted To Exposys Data Labs***

1. ***ABSTRACT***

Software-defined networking architectural framework eases the life of the network administrators by isolating the data plane from the control plane.

This facilitates easy configuration of the network, provides a programmable interface for developing applications related to management, security, logging etc. and the centralized logical controller gives more control over the entire network, which has the total visibility of the network.

These advantages of SDN also expose the network to the vulnerabilities and the impact of the attacks are much severe when compared to conventional networks, where the network devices in itself provided protection from the attacks and limits the scope of the attacks.

In this paper, we explore various attacks that can be launched on SDN at different layers. We also evaluate some of the existing security methods in mitigating the attacks. We also explore a possible solution to prevent DDoS attacks using entropy.

A Distributed Denial of Service (DDoS) attack is a DoS attack utilizing multiple distributed attack sources.

Every network in the system has an entropy. Increase in randomness causes decrease in entropy.

To mitigate this threat, this project shows how DDoS attacks can exhaust controller resources and provides a solution to detect such attacks based on the entropy variation of the destination IP address.

Based on this value if it drops below threshold , we are blocking the specific port in the switch and bring the port down.

This method is able to detect DDoS within the first five hundred packets of the attack traffic.

Keywords—SDN, security, threat vectors, networking, DDoS, Entropy, threshold, randomness, programmable, launch attack ,generate traffic.

***2. Table of Contents***

**1. ABSTRACT**

**2. Table of Contents**

**3. Introduction**

**4. Existing Method**

**5. Proposed method with Architecture**

**6. Methodology**

**7. Implementation**

**8. Conclusion**

1. ***INTRODUCTION***

Network security innovation is like a race between adversaries and the security communities to best each other in breaking and securing the network. Security researchers and practitioners have put much effort and made strides to overcome the threats posed by their adversaries. However, the rapid advancement of information and communication technologies, such as mobile devices and cloud computing virtualization, imposes an additional burden on network administrators in ensuring network security. Technology advancement also introduces new threats, attack methods, and attack vectors.

Why is SDN security important? We know SDN network has separated data plane and control plane unlike the conventional network architecture. This network architecture can be implemented as cloud service. Cloud services are exploding and big organizations and enterprise network administrators are migrating to the SDN-based network implementations. These virtual technologies provide predictability, manageability and good quality of service. In parallel with added advantage, importance of security provision in this centralized managed network has become one of the main concerns. With

the single centralized virtual server running as controller, which basically install and manages the flows in the data plane network agents through overflow communication protocol which makes the controller a primary victim for the attacker.

• Security standards implementation in overflow communication has not been defined and product developers are implementing their proprietary methods.

• The programmable aspect of Software Defined Networks also makes them more vulnerable to a number of malicious code exploits and attacks.

• The southbound interface of an SDN can also easily be targeted with diverse denial of service and side channel attacks.

• Configuration errors of SDN can have more serious consequences than in traditional networks.

• Establishing trust is crucial.

***PROBLEM STATEMENT***

With a listing of different security concerns in Software Defined Networks, one of the main security threats we are concentrating upon in this research work is on Distributed Denial-Of-Service.

When a large number of packets are forwarded to a network device with an intent to either stop the service or decrease the performance then such attacks are termed as Distributed Denial-of-service attacks.

In DDoS attacks, a large number of packets are sent to a host or a group of hosts in a network.

If the source addresses of the incoming packets are spoofed, which they usually are, the switch will not find a match and has to forward the packet to the controller.

The collection of legitimate and the DDoS spoofed packets can bind the resources of the controller into continuous processing that exhausts them.

This will make the controller unreachable for the newly arrived legitimate packets and may bring the controller down causing the loss of the SDN architecture.

Even if there is a backup controller, it has to face the same challenge

This kind of attacks can be detected at an early stage by monitoring few hundreds of packets based on the entropy changes.

The early detection of DDOS attack prevents the controller going down.

The term “early” is subjected to tolerance level and traffic being handled by the controller.

If detection happens early say first few hundreds of packet then, the impact of flooding of malicious packets can be controlled significantly.

The early detection mechanism must be of light weight and should have a high response time.

The high response time saves the controller in the period of attack to regain the control by terminating the DDOS attack.

1. ***Existing Methods***

***Principal Component Analysis (PCA)***  is a coordinate transformation method that maps the measured data onto a new set of axes, called the principal axes or components;

Each principal component points in the direction of maximum variation or energy remaining in the data, given the energy already accounted for in the preceding components.

So, the i th principal component captures the total energy of the original data to the maximal residual energy beside former i−1.

**Important terms**

**(a) OD Pair**

OD pair denotes a pair of nodes - the origin node and the destination node of one packet.

**(b) OD Flow p**

The OD flow consists of all traffic for an OD pair. If the network has k entrance, there will be k2 PoP pairs maximum, and hence k2 OD pairs. For short, the number of OD flows is set as p.

**(c) Number of successive time intervals of interest t**

Collect successive network’s traffic for a total of (w × t) seconds and separate the time period into t pieces => each time period = w seconds.

The number of time periods t can be adjusted to t1 by adjusting w to w1, so that

t × w = t1 × w1

1. ***Proposed Method***

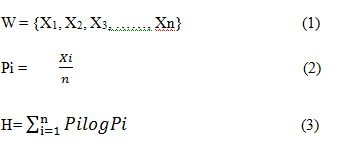
***Early Detection of DDoS Using Entropy***

***A. Measuring the Randomness***

The main reason for choosing entropy is its ability to measure randomness in a network. The higher the randomness the higher is the entropy and vice versa.

Let W be a set of data with n elements and X is an event in the set.

Then, the probability of X happening in W is shown (2).

To measure the entropy, referred to as H, we calculate the probability of all elements in the set and sum that as shown in (3).

The entropy will be at its maximum if all elements have equal probabilities.

If an element appears more than others, the entropy will be lower.

The size of W is called the window size.

If there is a continuous stream of incoming data, in our case the data is packet header, it will be divided into equal sets that are called windows.

In the window, each element and its occurrence are counted.

For instance, if the window has 64 elements and, all elements appear only once, the entropy will be 1.80. If one element appears 10 times, the entropy will be 1.64.

This property of entropy will be used for calculating the randomness in the SDN controller.

***B. Entropy for DDoS Detection***

Entropy is the method used in this research to detect DDoS attacks in SDN.

A look at the used methods in non-SDN networks is necessary before introducing it in SDN.

Since there is no research in using this method in SDN, we have to rely on what is done in non-SDN research.

There are two essential components to DDoS detection using entropy; window size and a threshold.

Window size is either based on a time period or number of packets.

Entropy is calculated within this window to measure uncertainty in the coming packets. To detect an attack, a threshold is needed.

If the calculated entropy passes a threshold or is below it, depending on the scheme, an attack is detected.

Qin et al. [2] propose a method with a window of 0.1 seconds and three levels of threshold. This method is concerned with avoiding false positive and false negatives in the network.

However, as the authors themselves mention, the method is time consuming and uses more resources.

Ra et al. [3] propose a faster way of computing entropy by basing the calculation on both packet type and the volume of packets in the network. This method also uses a time period window.

For the threshold, the authors ran several datasets to find a suitable threshold and it is a multiple of standard deviation of entropy values.

In this method, the false negatives are higher than other methods and false positives are lower. No percentage of accuracy is indicated.

There is also no mention of resources used for fast computation.

Entropy has been used in different ways to detect DDoS attacks in the network but, to the best of our knowledge, it has not been used in SDN.

In SDN, when passing packets to the controller, the limitation of available resources and the quick detection of attacks are key features of any detection scheme.

In this research, we will apply entropy for DDoS detection with the above limitation of the controller in mind.

***C. Statistics for Entropy Detection***

Before discussing our solution, we will look at an entropy-based DDoS detection method that is used in a non-SDN network.

Oshima et al. [1] propose a short-term statistics detection method based on entropy computation. “Short-term” here refers to calculating entropy in small size windows.

The study proposes a window size of 50 packets for gathering statistics. In this method, different window sizes were tested for optimal entropy measurement.

Table 2.1 shows the results of the tests for different window sizes.

**Table 2.1 Entropy of different window sizes [1]**

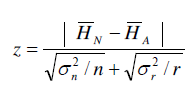
In Table 2.1, W is the window size, HN is the entropy in normal condition, HA is entropy during an attack, SN and SA are the standard deviation of entropy for normal and attack traffic conditions respectively.

Z is the test of significance. More precisely, it is a test of validity for the hypothesis between two averages of different populations.

When it is higher than 1.64, the hypothesis is valid.

In Table 2.1, it can be seen that for a window size of 50, z is 1.7.

The value can be computed using (4).



**(4)**

σn and σr are the same as Sn and Sa . n is the population of normal traffic packets (value of n is not given) and r is set to 25.

To test the hypothesis, a one-sided test of significance with 5% confidence interval was used.

The formula for the one-sided test is shown in (5) where x is the mean of the population,

μo is the sample mean, σ is the standard deviation and n is the sample count.



**(5)**

We have chosen the window size to be 50 for this research.

The main reason for choosing 50 is the limited number of incoming new connection to each host in the network.

In SDN, once a connection is established, the packets will not pass through the controller unless there is a new request.

The other reason is the fact that a limited number of switches and hosts can be connected to each controller.

The third reason for choosing this size is the computation that is done for each window.

A list of 50 values can be computed much faster than 500 and, an attack in a 50-packet window is detected earlier.

We also tested the entropy with three other window sizes and measured the CPU and memory usage.

Table 2.2 shows that there is no difference in memory usage but CPU usage increases with window size.

**Table 2.2 Window size comparison**

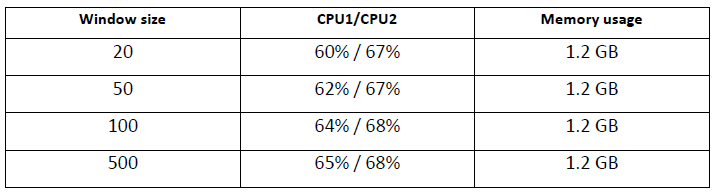


Table 2.3 shows the difference in entropy and the number of attack packets from each window size.

HN is the normal traffic entropy, HA is the attack traffic entropy and HN - HA is the difference. Last column shows the number of malicious packets when the attack traffic is 25% of all incoming packets.

This is the lowest attack traffic rate that our method can detect with accuracy.

In a window of size 20, the difference of entropies is less than 10% making it difficult to choose a threshold.

With only five packets, probabilities of false positives will increase. On the other side, window of 500 does not offer a better difference of entropies and takes a much longer time than a window of 50 to compute entropy.

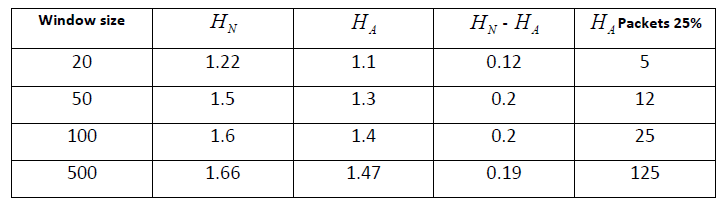
The difference is 0.19 for 500 packet window size which is 11% drop in normal traffic entropy. The difference in the window size of 50 is 0.12 which 10% drop in normal traffic entropy.

Difference of 1% does not justify choosing a 10 times bigger window size.

Window sized of 50 and 100 look close. Because the number of hosts in our test network is less than 100, we chose 50.

It is very easy to change window size in the controller and this flexibility is the advantage of SDN.

**Table 2.3 Comparison of five windows**



***D. Early Detection in OpenFlow Controller***

As it was shown before, one function of the controller is collecting statistics from all OpenFlow switches to detect inactive flows. These flows will be removed if they do not receive any packets for a period of time.

This time period is called time-out in the OpenFlow specification and it can be set to different values.

For a lightweight solution, we propose adding another set of statistics to the controller. In this work, it is the entropy of the destination IP address in the controller. The function will determine if a higher-than-normal rate of incoming packets destined to the same destination.

In the previous section, the window size was set to be 50. The assumption is that the network has 50 or more hosts connected to it.

One other component of DDoS detection by entropy is selecting an appropriate threshold, which will be discussed in the next chapter.

In the new function, every 50 Packet In messages will be parsed for their destination IP address and the entropy of the list will be computed.

The calculated entropy, then, will be compared to a threshold. If the calculated entropy is less than the threshold and it persists for a minimum of 5 consecutive entropy periods, it will be considered an attack.

Detection within 5 entropy periods is 250 packets in the attack, which gives the network and early alert of attack.

We tested with values one to five consecutive periods and five has the lowest false negative and positive for early detection.

With a window of 50 packets and a network of 50 hosts or more, maximum entropy, HMAX is when each of the 50 packets is equally distributed among all the hosts.

When an attack happens, the number of packets going to the same destination host, or the same subnet, is much higher so it will make the target unreachable to legitimate traffic.

This would be the main objective of the attack. The other factor in an attack is its flow to the target.

The attack packets will be targeting a single host or a subnet. If the rate of attack to a host is higher than the normal traffic level, which is always the case, the number of packets to that particular host in a window will increase.

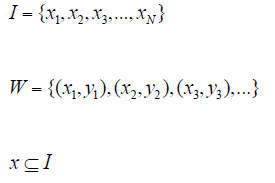
Because of that, the entropy will fall with a certain percentage. If it falls below the threshold, it is an attack.

One advantage of our method is the liberty of testing the controller with different attack rates to calibrate a threshold.

In SDN, the controller can be connected to a simulator and tested then be deployed in the field to accept production network traffic.

This property allows for the threshold to be tested before applying it.

Let I be the IP addresses of all hosts connected to the network, see (6), and W be the window containing new packets’ destination IP address x and their number of occurrence y , in (7).

 **(6)**

**(7)**

**(8)**

Then the entropy will be at its maximum if each destination IP is unique.



**(9)**

If the above condition does not hold, then some IP addresses have appeared more than once.

Two conditions are chosen to be the trigger for an attack in our method. One is the threshold and the other is the continuity of the attack.

There might be glitches in the network that cause irregularity in normal traffic.

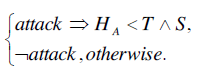
If a link to a switch goes down or some hosts become temporarily unavailable, the entropy might fall and trigger a false positive.

To avoid false positives of this type, we propose a limit for the number of consecutive low entropy windows.

Based on that, the condition for declaring an attack is shown in (10) where T is the threshold, S is an array of five windows with lower than T entropy. “-” is the sign of negation.

An attack happened if entropy of attack HA, is smaller than the threshold and, having five consecutive lower than threshold entropies is true.

Otherwise, there is no attack:



**(10)**

1. ***METHODOLOGY***

***• Flow Aggregation***

Flow is a unidirectional series of IP packets of a given protocol travelling between a source and a destination IP/port pair within a certain period of time. Flow aggregation techniques are used to aggregate flows into a single flow with a larger granularity of classification giving a flow count for each connection with a unique combination of attributes given below for a packet.

**– Source IP**

**– Destination IP**

**– Source Port**

**– Destination Port**

**– Protocol**

Aggregated flows have a larger number of packet information that dramatically reduces the amount of monitoring data.

Hence, Internet traffic flow profiling has become a useful technique in the passive measurement and analysis field.

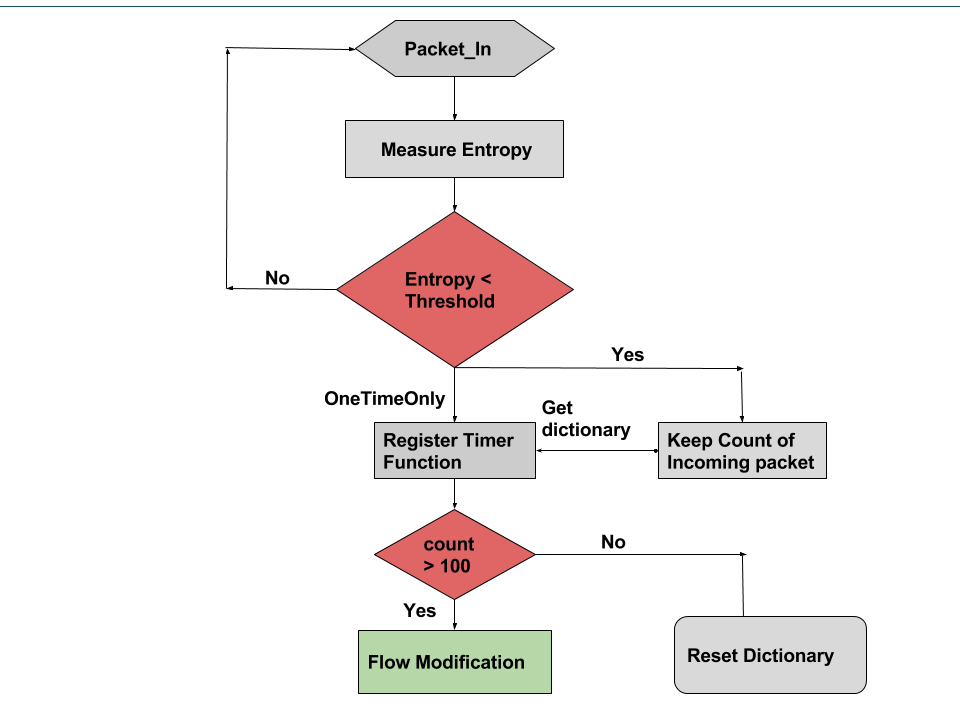
Instead of considering the packet count of each connection, the proposed method calculates the flow count of each connection at particular time interval for detecting the flooding attacks, increasing the speed of analysis and reducing the time complexity of our algorithm.

***• Entropy Approach***

In this detection approach, entropy of flow count is calculated for each connection using the formula given below.

**Hi,t = −log(xi,t/(x1,t + x2,t + ... + xn,t)) + Ri,t**

**If (xi,t ≥ xi,t+1) then r ← |log(xi,t+1/xi,t)| else r ← |log(xi,t/xi,t+1)|.**

***Figure 1: Algorithm Flowchart***

It is essentially a standard way of calculating any entropy in a system.

When there is an attack, entropy drops drastically, because there is one flow count that is dominating.

In the non-attack case, the entropy will be in a constant range.

Let a random variable xi,t represent the flow count of a particular connection i over a given time interval t.

***• Adaptive Threshold Algorithm***

In flooding attack detection, threshold value is very important.

Threshold value needs to be updated according to the packet traffic condition.

On one hand, if an attacker sends malicious traffic with small change in traffic when the channel is stable, the detector cannot detect the attack with high value of B, where B is the threshold multiplication factor.

Because of the steady channel condition and stealthy attack pattern, the detection facility does not work properly with highly set B.

On the other hand, if the channel is burst but the detector has small B, the detector works very sensitively in this situation.

As a result, the detector yields many false positives, which are not severe but a bad characteristic of the detector.

In the proposed method B value is updated based on entropy value. The detection algorithm is shown in figure below. The B will be changed under the following rules:

**If (Hi,t > 1.5ut) then B ← B + 1**

**If (Hi,t < 0.5ut) then B ← B − 1**

Hi,j refers to the fast entropy, ut and st is the mean and standard deviation of flow count during a particular time interval.

Di,,t is defined as the difference between the mean value ut and the fast entropy Hi,t.

While applying adaptive threshold algorithm, if Di, t is greater than the product of B and s indicates flooding attack.

1. ***Implementation***

# 1. Pre-Requirements for the experiment setup , Links and Manual to install.

## a. Mac OS, Windows with Ubuntu Operating System.

The following experiment setup works better for almost all kinds of operating systems.

## b. Virtual Box or VMware virtual player

* Virtual box can be downloaded using the following link: <https://www.virtualbox.org/wiki/Downloads>

## c. Mininet network Emulator

* Ensure suitable version of Virtual Box is installed into your system.
* Download suitable version of mininet using the following link:

http://mininet.org/download/

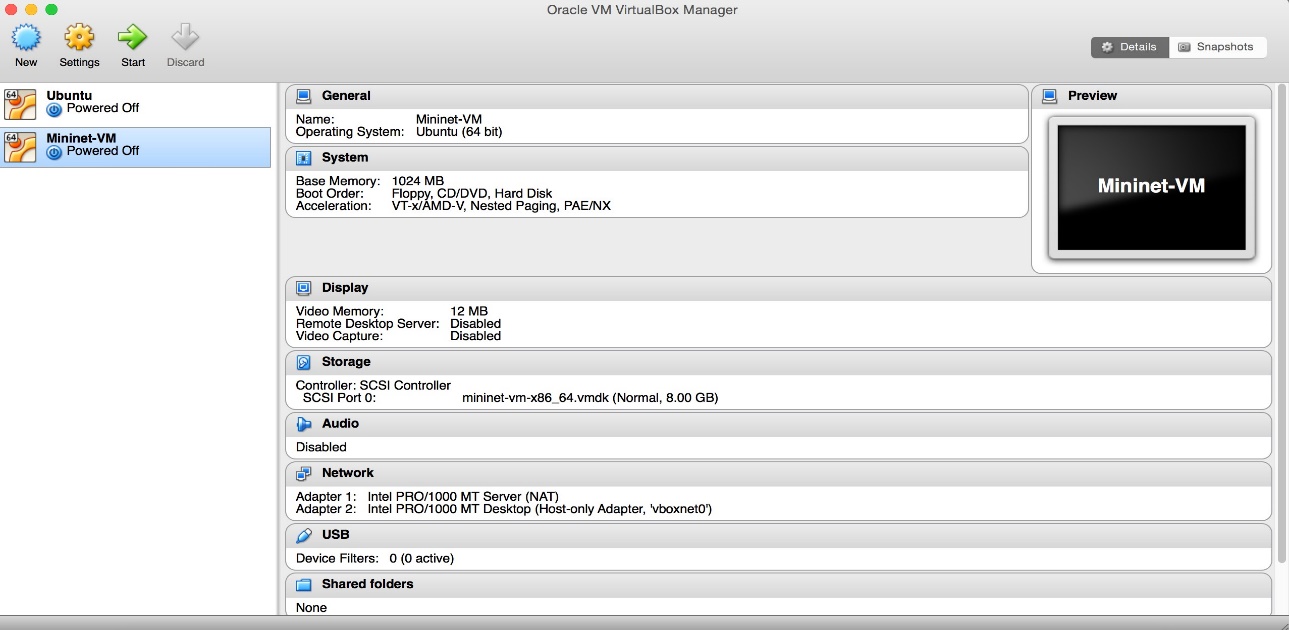
* Open the virtual Box and click on import Appliance.
* Search for a ‘.ovf ’ extension of mininet and click on Ok.
* This process should install Mininet on the Virtual Box
* On successful installation one can able to see the mininet on the virtual box as shown in the Figure-­‐ 1 below: 

Figure 1: Virtual machine

## d. Installation of Python

* Type the following command to check if python is installed on mininet:

$ python –V

* The above command should display the version of python.
* If python is not installed on the mininet, type in the following command to install it manually:

$ sudo apt-get install python

## e. Installation of Scapy

* Most recent versions of mininet comes with pre-­‐ installed versions of python and scapy
* To verify whether scapy is installed type the following command logging into mininet:

$ sudo scapy

You should be able to see an output as shown in the figure below:

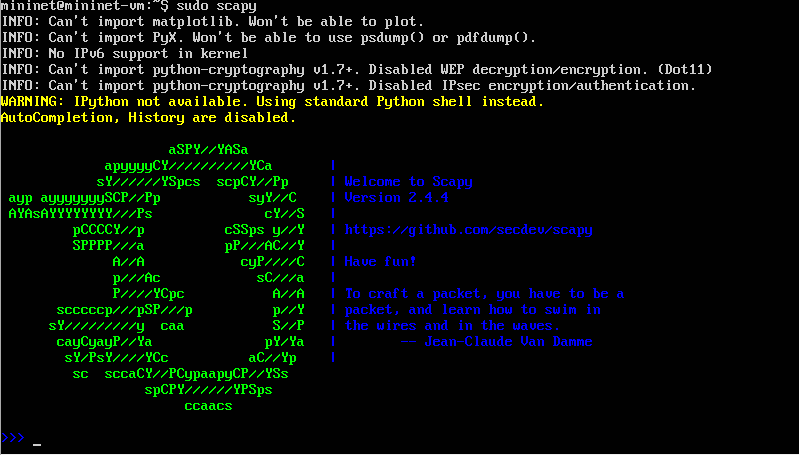


Figure 2 : Scapy installation

* The above figure shows that the scapy has been pre-­‐ installed. If an error pops up, scapy has to be pre-­‐ installed manually.
* Type quit() to exit from scapy
* To install scapy manually type in the following command:

sudo apt-get install python-scapy

# 2. Creating the Test Environment.

a. Do a ssh to mininet from your local host giving the command:

$ ssh –X mininet@IPAddress

b. Create a new python script for normal traffic generation in the folder mininet/custom

$ cd mininet/custom

$ vim launchTraffic.py

c. Copy the contents of script traffic.py provided in the appendix of the manual and save the file.

d. Similarly repeat Step b and c to create an attack traffic file

$ vim launchAttack.py

e. Now go to the mininet in the virtual box and run the following commands:

$ cd pox/pox/forwarding

$ vim detection.py

f. Copy the contents of the detection.py script provided in the appendix into above file and save it.

g. now make copy of the l3\_forwarding module using the following command:

$ cp l3\_forwarding.py l3\_edited

Note: The above step can omitted if you prefer to edit the already existing pre-­‐installed l3\_forwarding module.

h. Once a copy is made, you must be able to see the same contents of l3\_forwading.py file in your newly created file. Now modify the file as shown in appendix and save it.

# 3. Steps to perform the experiment.

## Find the Threshold for usual Traffic(normal)

a. On successful setting up of the testbed, you should able to see a ssh window of mininet and mininet running on the virtual box as shown in the figure below:

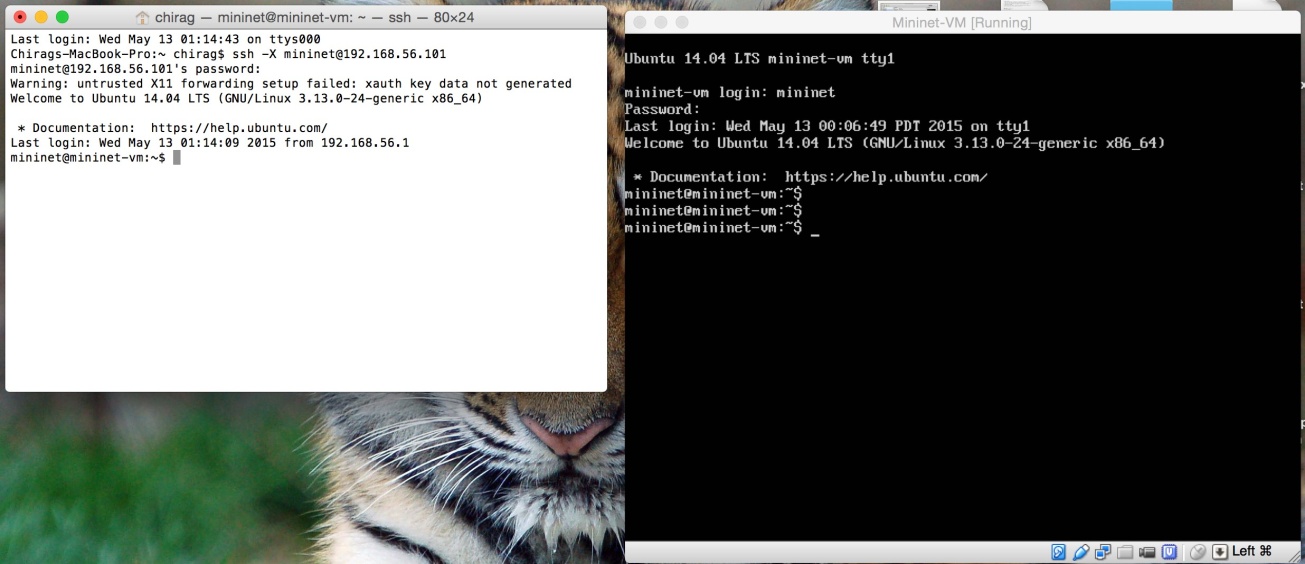
Change the image

Figure 3:ssh the mininet

b. In the mininet terminal of virtual box enter the following command to run the pox controller:

$ cd pox

$ python ./pox.py forwarding.l3\_edited

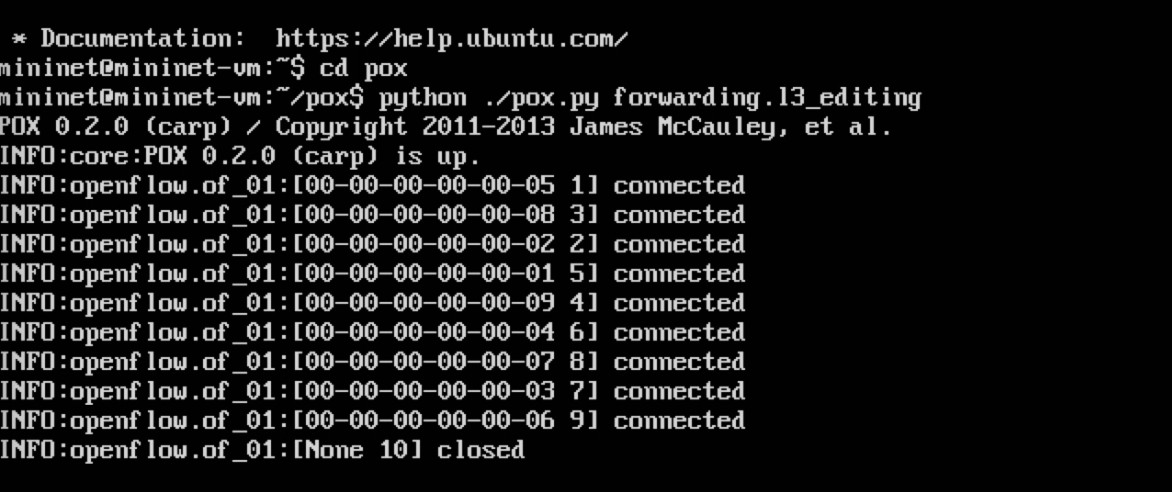


Figure 4: Installing Pox controller

c. Now create a mininet topology by entering the following command in the ssh window:

$ sudo mn --switch ovsk --topo tree,depth=2,fanout=8 -- controller=remote,ip=127.0.0.1,port=6633

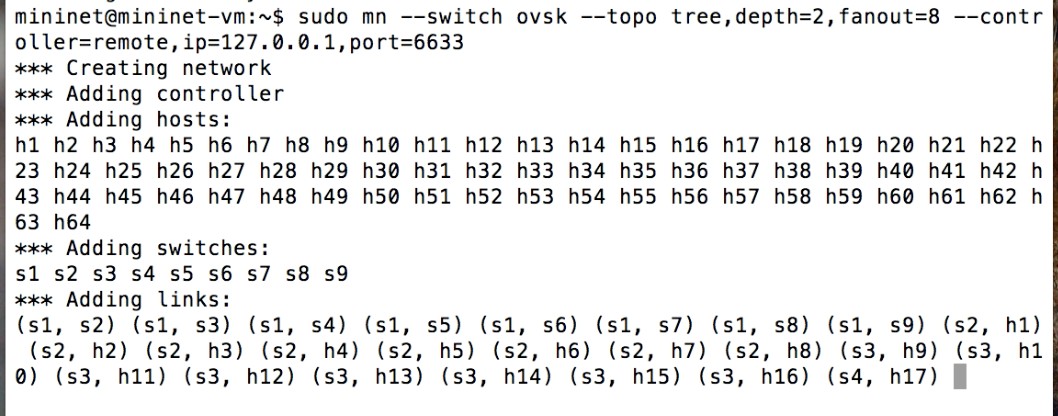


Figure 5:Creating the topology

Note: To determine the IP address of your pox controller

Enter the following command in the mininet of your virtual box:

$ ifconfig

The loopback address is your IPaddress in the above command.

d. Now open xterm for an host by typing the following command:

mininet>xterm h1 h2 h3 h64

e. In the xterm window of h1 run the following command:

# python traffic.py –s 2 –e 65

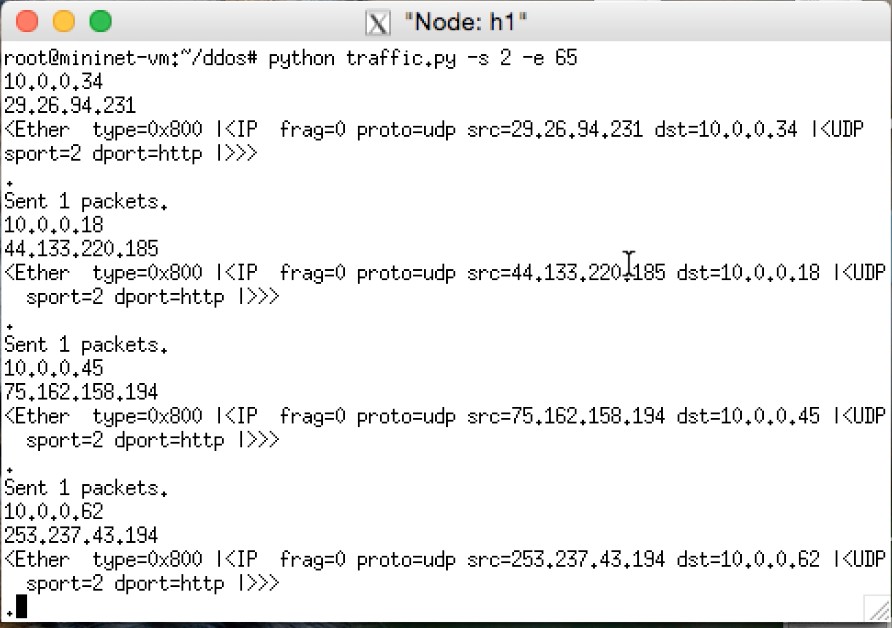


Figure 6:Generating the traffic

f. Now you should be able to see the pox controller generating a list of values for entropy as shown in the figure below. The least value obtained is the threshold entropy for normal traffic. To avoid false positives and negatives due to loss of a switch we choose an entropy value as 1.00 instead of 1.14. This implies 10% fault tolerance.

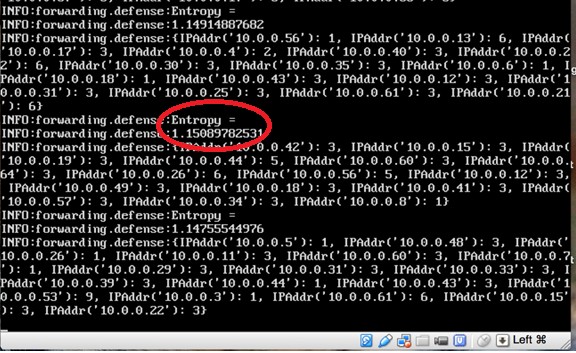


Figure 7: Entropy value

## II. Detection of DDoS threat using the value of Entropy

g. Now on xterm window of h64 enter the following commands:

# script h64.txt

# tcpdump –v

h. Now repeat step e on h1 and parallelly enter the following commands to run the attack traffic on h2 and h3 xterm windows:

# python attack.py 10.0.0.64

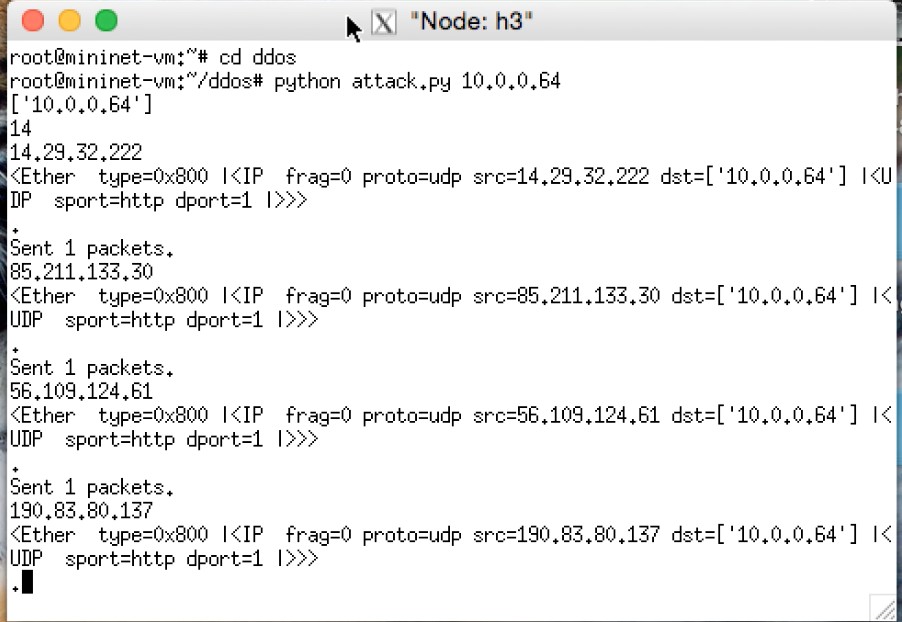


Figure 8: Launching the attack

i. Observe the entropy values in the pox controller. The value decreases below the threshold value for normal traffic as determined in step f (shown in the figure below). Thus we can detect the attack within the first 250 packets of malicious traffic attacking a particular host in the SDN network.

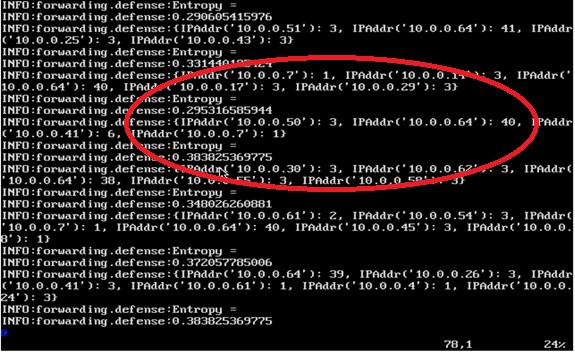


Figure 9:Change in entropy value of controller

j. On successful completion of experiment, terminate tcpdump on h64 by entering ‘control/command + c’

k. Stop running mininet topology by entering the following command:

mininet>exit

1. ***Conclusion :-)***

An efficient DDoS attack detection method using entropy approach was designed and implemented.

The flow count is calculated for each connection at particular time interval.

From the observation itis clear that the fast entropy value is considerably reduced for particular connection and particular time interval of which flow count is large value compared to rest.

DDoS attack is detected, when the difference between entropy of flow count at each instant and mean value of entropy in that time interval is greater than the threshold value.

Since the threshold value is updated adaptively based on traffic pattern condition, the accuracy of detection is improved

***THANK***

***YOU***