

CO-368
Internet Technology and Applications

**Analysis of DDoS Attacks in SDN
Environments**

Team members

Aswanth P. P. (15CO112)

Mohammed Ameen (15CO131)

Joe Antony (15CO220)

Table of Contents

| | |
|---------------------------------|----------|
| Table of Contents | 1 |
| Overview | 2 |
| Literature Survey | 3 |
| References | 3 |
| Methods Used | 3 |
| 1. Sample Entropy | 3 |
| 2. Principal Component Analysis | 5 |
| Implementation | 7 |
| Steps To Reproduce the Result | 7 |
| Prerequisites | 7 |
| Initial Steps | 7 |
| Detection Using Sample Entropy | 8 |
| Detection Using PCA | 8 |
| Conclusion | 9 |

Overview

In this project, we are trying to analyse DDoS attacks on SDN environments based on the paper **“A Novel DDoS Attacks Detection Scheme for SDN Environments”** by Di Wu, Jie Li, Sajal K. Das, Jinsong Wu and Yusheng Ji §.

A Denial-of-Service (DoS) attack is a cyber-attack where the attacker seeks to make a machine or network resource unavailable to its intended users by temporarily or indefinitely disrupting services of a host connected to the Internet. This is typically accomplished by flooding the target with superfluous requests in an attempt to overload systems. In a Distributed Denial-of-Service (DDoS) attack, the incoming traffic flooding the victim originates from many different sources. This effectively makes it impossible to stop the attack simply by blocking a single source.

Sample entropy, a general method for DDoS detection in SDN, is conducted by collecting the flow statistics or traffic features from the switches, and calculating the entropy measure randomness in the packets that are coming to a network. The higher the randomness, the higher is the entropy and vice versa. By setting a threshold, if the entropy passes it or below it, depending on the scheme, an attack is detected.

Principal Component Analysis (PCA) is a coordinate transformation method that maps the measured data onto a new set of axes. These axes are called the principal axes or components, where each principal component has the property that it points in the direction of maximum variation or energy remaining in the data, given the energy already accounted for, in the preceding components.

Literature Survey

We explored a variety of references to implement the detection of DDoS attacks via Sample Entropy [1][2] and Principal Component Analysis [3][4][5]. Previous studies of quick DDoS detection [6][7][8], show some treatment against DDoS attacks on SDN, but there are shortages in these work. For example, [8] used the entropy to describe the traffics, which may cause a false alarm when traffic feature getting larger. [6] only studies the flows with low traffic, which is powerful to solve the traffic pattern, which cannot handle other kinds of DDoS attacks.

References

1. [Early Detection of DDoS Attacks in Software Defined Networks Controller](#) (Seyed Mohammad Mousavi)

2. [An Entropy-Based Distributed DDoS Detection Mechanism in Software-Defined Networking](#) (Rui Wang, Zhiping Jia, Lei Ju)
3. [Mining Anomalies Using Traffic Feature Distributions](#) (Anukool Lakhina, Mark Crovella, Christophe Diot)
4. [Diagnosing Network-Wide Traffic Anomalies](#) (Anukool Lakhina, Mark Crovella, Christophe Diot)
5. [Structural Analysis of Network Traffic Flows](#) (Anukool Lakhina, Konstantina Papagiannaki, Mark Crovella, Christophe Diot, Eric D. Kolaczyk, and Nina Taft)
6. [A detection method for a novel ddos attack against sdn controllers by vast new low-traffic flows](#) (P. Dong, X. Du, H. Zhang, and T. Xu)
7. [Lightweight ddos flooding attack detection using nox/openflow](#) (R. Braga, E. Mota, and A. Passito)
8. [Early detection of ddos attacks against sdn controllers](#) (S. M. Mousavi and M. St-Hilaire)

Methods Used

1. Sample Entropy

Sample Entropy is a method used to detect DDoS attacks in SDN. 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.

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 in Equation 1. To measure the entropy, referred to as H , we calculate the probability of all elements in the set and sum that as shown in Equation 2.

$$W = \{ x_1, x_2, x_3, \dots, x_n \}$$

$$p_i = x_i / n \quad \text{--- (1)}$$

$$H = - \sum_{i=1}^n p_i \log p_i \quad \text{--- (2)}$$



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, it will be divided into equal sets that are called windows. In the window, each element and its occurrence are counted.

When packets arrive at the controller, the source address is always new. There has not been an instance of them in the table of the switch so they are passed on to the controller. For every new incoming connection, the controller will install a flow in the switch so that the rest of the incoming packets will be directed to the destination without further processing. Hence, any time a packet is seen in the controller, it is new.

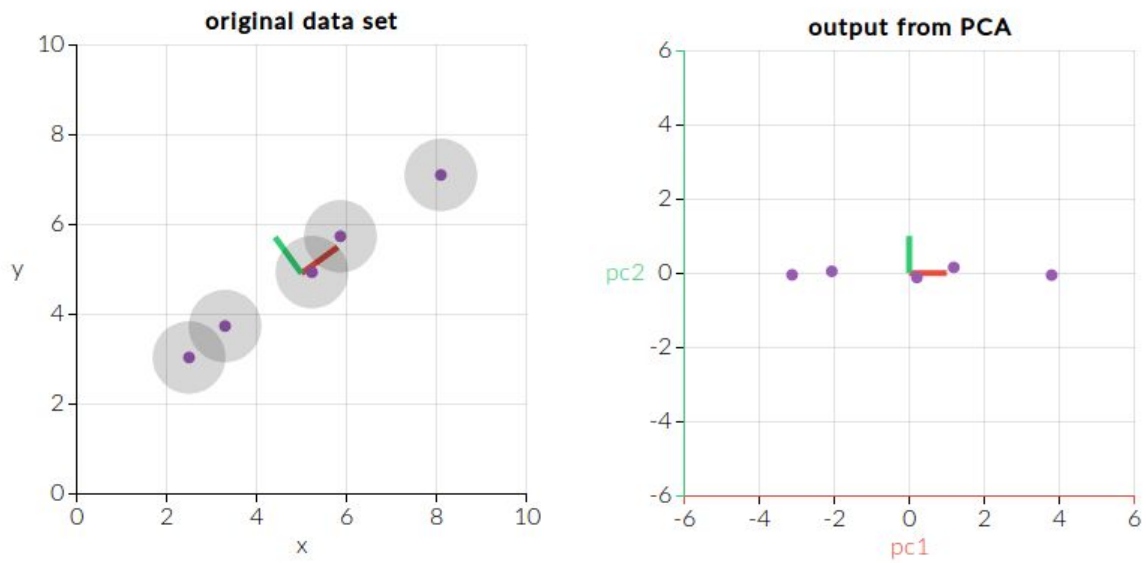
The other known fact about the new packets coming to the controller is that the destination host is in the network of the controller. The network consists of the switches and hosts that are connected to it. Knowing the packet is new and the destination is in the network, the level of randomness can be quantified by calculating the entropy based on a window size. The window size is the number of incoming new packets that are used for calculating entropy. In this case, maximum entropy occurs when each packet is destined to exactly one host. Minimum entropy occurs when all the packets in a window are destined for a single host.

Being able to quantify randomness and have minimum and maximum based on entropy makes it a suitable method for DDoS detection in SDN. Using entropy, it is possible to see its value drop when a large number of packets are attacking one host or a subnet of hosts.

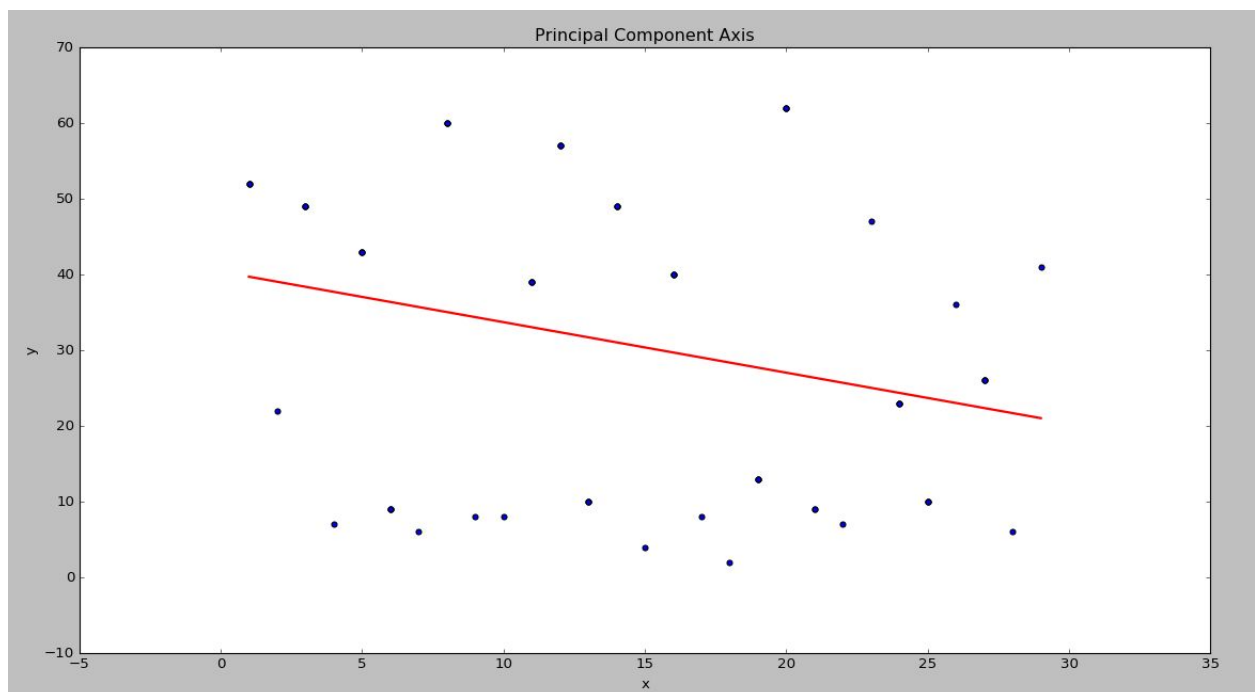
2. Principal Component Analysis

Principal component analysis (PCA) is a mathematical procedure that transforms a number of (possibly) correlated variables into a (smaller) number of uncorrelated variables called principal components. The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible.

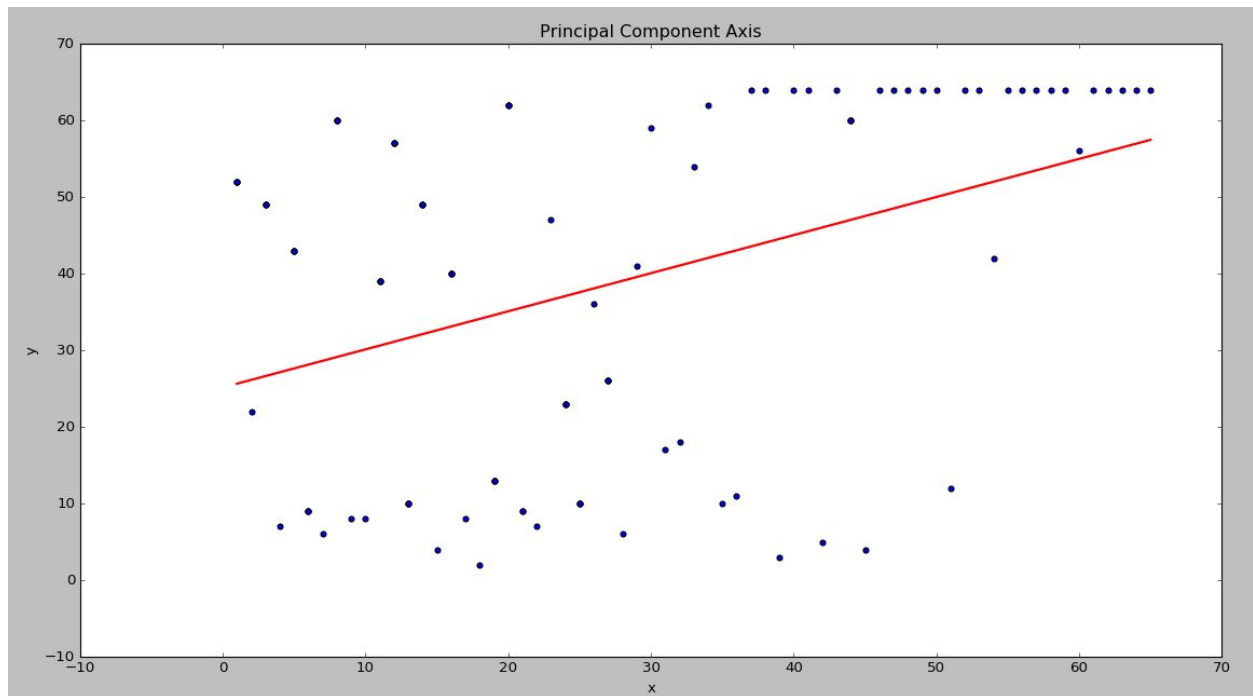
PCA is used to emphasize variation and bring out strong patterns in a dataset. It's often used to make data easy to explore and visualize. For example, consider a dataset in only two dimensions, say (x, y) . This dataset can be plotted as points in a plane. But if we want to tease out variation, PCA finds a new coordinate system in which every point has a new (x', y') value. The axes are combinations of x and y called "principal components" that are chosen to give one axis lots of variation.



PCA axis while normal traffic:



PCA axis while attack:



Implementation

Steps To Reproduce the Result

Prerequisites

1. Install Python.
 2. Install mininet along with pox controller:
 - (a) mininet installation : <http://mininet.org/download/>
 - (b) pox controller
- Clone the repository : <http://github.com/noxrepo/pox>

Initial Steps

1. Clone the repo:

<https://github.com/aswanthpp/Analysis-of-DDoS-Attacks-in-SDN-Environments>

2. Copy the following files from src folder:

(a) Packet-Generation/traffic.py

(b) Packet-Generation/attack.py

to [mininet/custom](#)

(a) POX-Detections/detectionUsingEntropy.py

(b) POX-Detections/detectionUsingPCA.py

(c) POX-Detections/l3_detectionEntropy.py

(d) POX-Detections/l3_detectionPCA.py

to [pox/pox/forwarding](#)

Detection Using Sample Entropy

1. Run the pox controller:

```
$ cd pox
```

```
$ python ./pox.py forwarding.l3_detectionEntropy.py
```

2. Create a mininet topology by entering the following command in another terminal:

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

3. Open xterm for the following hosts:

```
mininet>xterm h1 h2 h3 h64
```

4. In the xterm window of h1, run the following commands to launch traffic:

```
$ cd ../mininet/custom
```



```
$ python traffic.py -s 2 -e 65
```

5. Now the pox controller generates a list of values for entropy. The least value obtained is the threshold entropy for normal traffic.

6. Repeat step (4) on h1 and parallelly enter the following commands on h2 and h3 xterm windows to launch the attack:

```
$ cd ../mininet/custom
```

```
$ python attack.py 10.0.0.64
```

Observe the entropy values in the pox controller. The value decreases below the threshold value for normal traffic. Thus we can detect the attack.

Detection Using PCA

1. Run the pox controller:

```
$ cd pox
```

```
$ python ./pox.py forwarding.l3_detectionPCA.py
```

2. Create a mininet topology by entering the following command in another terminal:

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

3. Open xterm for the following hosts:

```
mininet>xterm h1 h2 h3 h64
```

4. In the xterm window of h1, run the following commands to launch traffic:

```
$ cd ../mininet/custom
```

```
$ python traffic.py -s 2 -e 65
```

5. Now the pox controller generates a list of values for deltaY, which is the difference between the y-coordinates of a packet and the point obtained by drawing a perpendicular from this packet to the Principal Component Axis.

6. Repeat step (4) on h1 and parallelly enter the following commands on h2 and h3 xterm windows to launch the attack:

```
$ cd ../mininet/custom
```

```
$ python attack.py 10.0.0.64
```

Observe the deltaY values in the pox controller. The values converge to the interval (-1, 1). Thus we can detect the attack.

Conclusion

Some of the key ideas of SDN are the introduction of dynamic programmability in forwarding devices through open southbound interfaces, the decoupling of the control and data plane, and the global view of the network by logical centralization of the “network brain”. While data plane elements became dumb, highly efficient and programmable packet forwarding devices, the control plane elements, are now represented by a single entity, the controller.

A Distributed Denial of Service (DDoS) attack is an attempt to make the services of network unavailable by exhausting the resources. The effect of this DDoS attack is even worse in an SDN than the traditional one. This is because, the controller will be invoked every time for a new [O,D] pair. Hence within a short span of time if the attacker bursts too much packets into a network it will damage the controller, followed by the collapse of the network.

We have implemented two methods to detect DDoS attacks successfully. The results show that PCA is a better approach than the sample entropy because:

In sample entropy, every packet which has entropy value less than the threshold value, is stored in a dictionary and if an entry comes more than 5 times, it is assumed as a DDoS attack. This knob value of 5 depends on the topology. Hence, a topology smaller in size may show DDoS attack for normal traffic also.

But in PCA analysis, we are taking characteristics of each packet to update the principal component axis. Each packet destination value is compared with current principal component axis. During an attack, the principal component axis is gradually shifted towards the destination value. Hence, there will be successive decrease in deltaY values, which is an indication of DDoS attack.