Verifying the Suspicious Port in Case of Abnormal Behavior in Software Defined Network

Muhammad, Mamdouh
Communication Networks Group
MSCSP Degree Course
P.O. Box 100565, D-98684 Ilmenau, Germany
mamdouh.muhammad@tu-ilmenau.de

Soliman Awad Alshra´a, Abdullah
Communication Networks Group
MSCSP Degree Course
P.O. Box 100565, D-98684 Ilmenau, Germany
Abdullah.Alshraa@tu-ilmenau.de

***Abstract*—Software-Defined Networks (SDN) as technology of isolating data plane from control plane as shown in Fig. 1 has attracted many researchers in universities and companies. In SDN, network engineers can easily manage and troubleshoot the whole network infrastructure due to the centralization of the control plane. SDN offers many advantage over traditional network which ease network automation through using application programming interfaces (APIs).**
Distributed Denial of Service (DDoS) attack is popular attack type, in which many attack sources (Botnets) are being used to launch attack towards specific device to affect its availability.
Entropy value is how much information the result is containing and it is considered as a measure of randomness. The higher the randomness of output, the higher the entropy value. [10]
Using Entropy-based algorithm to detect DDos attacks in SDN is the objective of this paper.

***Index Terms*—SDN, DDos, Entropy**

I. Introduction

DDos attacks are one of the main severe attacks that can affect badly the availability of many important institutions that depends mainly on running servers such as banks.One of the most biggest attacks that ever happened was a DDos attack happened to github website. In which, a traffic of 1.35 Tbps affected the github availability for 10 min. [1]. Also in 2019 in Germany there was a DDos attack that affected Wikipedia website availability [2]. DDos attacks in SDN technology is an important hot topic for researchers to investigate. Flooding of traffic from botnets to a victim to affect its performance or make it unavailable is the main goal of DDos attacks.
The existence of many DDos attacks types makes it a huge challenge for researchers to find a mitigation foe every DDOS attack type. For example, the main three DDos attacks based on previously occured cases are TCP flood attack, ICMP flood attack and UDP flood attack [8]. Other DDos attacks are Domain Name System (DNS) attack and Hyper Text Transfer Protocol (HTTP) attack.
Entropy as an unique measure of occurrence possibility [10] for different network parameters is one of two main methods used to detect DDos attacks in SDN networks. The main two methods being used are statistical analyses [10] in which mostly entropy value is being used and machine learning in which we can train our algorithm to automatically modify parameters based on using trained data.
This paper is organized as following: in section 2 we will refer to some of past related papers, Section 3 giving more information about Entropy value as a solution to detect DDos attacks, Section 4 will introduce our algorithm to detect and mitigate DDos attacks, section 5 we present our simulation results and then in section 6 conclude our paper and future work.

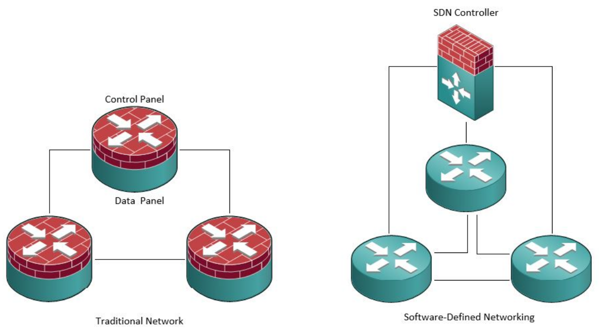


Figure 1. Traditional network vs SDN

II. Related previous works

In [8],[10],[9] works, they proposed Entropy-based algorithms to detect DDos attacks. M. Kia In [8] used POX controller and Mininet to detect DDos attacks using Entropy variation and the flow rate.
Mousavi in [10] proposed an effective way to detect DDos attacks in early stages using also Mininet to generate and emulate tree topology and POX controller. He proposed that his approach will detect DDos attacks within the first 500 attack traffic packets.
Fatima in [9] proposed an Entropy-based algorithm to detect DDos attacks in 64 hosts fat topology.
In [**IDS**],[12] works, they proposed machine learing algorithms to detect DDos attacks in SDN. In [**iii**] they designed new IDS based on many machine learning algorithms such as Naive Bayes, K-Nearest neighbour and K-means to differentiate between benign and attack traffic. In [12], they used Neural Network (NN) and SVM classifiers to detect DDos attacks in

SDN.
Pomona in [13], investigated detection of DDos attacks in SDN using entropy value using Raspberry Pi 3 to convert it into OF switch.
Carvalho and others in [14] used entropy value also to detect TCP-SYN flood attacks and they proposed different window size.
Li and others in [15] prposed lightweight DDos attacks early detection scheme in SDN based on $\phi - entropy$.

As shown in fig. 3, we used Mininet [16] as an emulator for SDN networks, and to plot this topology, we used Edraw diagramming tool [3]. In this figure, we can see the SDN network contains one controller, four OpenFlow switches, and four hosts. As depicted and mentioned earlier, if C1 - the controller goes down, the whole system will breakdown.

III. Entropy value

As mentioned before, statistical methods are one of two methods being used in detecting DDos attacks. Mainly in DDos attacks detection statistical methods we use entropy value as we use as a measure of randomness in network traffic. Entropy is directly proportional to randomness.The main two components we use in our entropy solution is Window size and threshold. Window size is the total number of coming new packets and i our case we set it to be 50,300 and 500 new packets as shown in table I.

A. *Calculating threshold*

Threshold is the value below it we consider there is an attack and it is equal to around 1.1, 0.27 and 0.1 in our case and due the change of window size packets number, we must calculate the corresponding suitable threshold [11]. Using the following eq.(4) we can determine the new threshold in every case.

$$R = \frac{P_a}{P_n} * 100\% \tag{1}$$

Where **R** is the resultant rate, **Pa** represents attack packets number and **Pn** is benign packets number.

And as we can see from fig. 2, The higher the number of window size, the lower the entropy value, the smaller the difference between the entropy value and the threshold, the higher computation needed (CPU load), the later the attack can be detected. We are calculating entropy based on measuring the randomness in the traffic using destination IP address. For every new coming packet, we are using the following eq.(1) to calculate the destination IP and how many times it occurred.And to measure the provability, we use eq.(2). After we successfully receive our window size totally, which is 50 - default value for our work - in our case, we use eq.(3) to calculate the entropy value (H)

$$W = (X_1, Y_1), (X_2, Y_2), (X_3, Y_3), ..., (X_n, Y_n) \tag{2}$$

$$P_i = \frac{X_i}{n} \tag{3}$$

$$H = -\sum_{i=1}^{n} P_i \log P_i \tag{4}$$

Table I
Window size, Entropy value and Threshold

Window Size	Entropy Value	Threshold
50	1.1	0.54
300	0.317	0.129
500	0.21	0.1

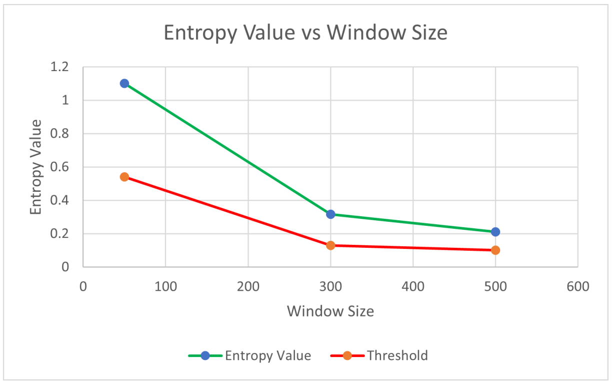


Figure 2. Window size and corresponding entropy values

Where **W** is a group of data with total n components, **X** represents an incident in **W**, **Y** is number of repetitions of this incident and **Pi** is the probability of occurrence of this incident. As mentioned above, to have max entropy, it is supposed that every packet is destined to different IP. If many packets are destined to the same IP, entropy value will be reduced and if it go below the threshold value, it will be considered as indication of an Attack. We assumed a window size of 50 - default value for our work - packets and we will propose five continuous window size packets number that go below the threshold to declare it as an attack.

IV. Entropy-based DDos attack detection method

In order to emulate the behaviour of DDos attacks in SDN networks, we are using Mininet which is a software to emulate Openflow switches behaviour[16].
For SDN controller, we use POX controller [4] which by default is being installed along with mininet. POX controller is developed version of NOX controller. POX controller is a python-based open-source cross-platform controller.
For crafted packets generation, we are using Scapy. Scapy is python-based program to craft and generate packets [5].

We implemented the entropy detection algorithm using a laptop with following hardware specifications mentioned in Table II,

Regarding software specifications, they are mentioned in Table III,

Table II
LAPTOP SPECIFICATIONS TO EMULATE THE ALGORITHM

	CPU	RAM	Hard Disk
value	I7 4910QM	16G	512 SSD

Table III
SOFTWARE SPECIFICATIONS TO EMULATE THE ALGORITHM

	OS	Emulation	Controller	Scapy
version	UBUNTU 20.04	Mininet 2.2.2	POX	2.4.4

Regarding our topology, as shown in fig. 3, we are proposing a fat tree topology that contains 1 controller 4 OvS switches and 16 hosts. All proposed generated benign traffic or attack traffic is being launched via host 1 (10.0.0.1). The host to be attacked in our approach is host 16 (10.0.0.16).

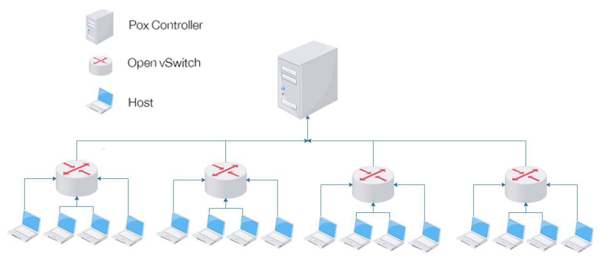


Figure 3. Proposed Topology

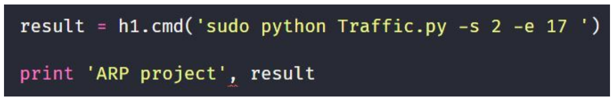
To generate benign and attack traffic, two python scripts are being used and launched in host 1.
Benign traffic python file is generating random traffic usind "randrange" function from random ip range excluding previously reserved IPs like Automatic Private IP Addressing (APIPA) IPs range (169.254. 0.1-169.254. 255.254), local host IPs range (127.0.0.0 to 127.255.255.255) and RFC 1918 private IPs. The destined IPs for this benign traffic are from host 2 till host 16.

Attack python file is generating traffic from same random IP ranges as in the benign traffic python file, but the difference here is that we are using host 16 (10.0.0.16) as the only destination host.

To accurately detect entropy value in all possible cases, we propose three phases.

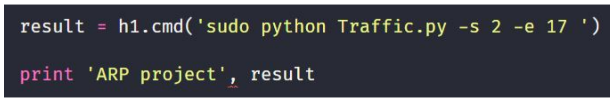
1- ***Phase 1*** as shown in 4 Only benign traffic is generated and destined to range of hosts.
In this case entropy value will be around 1.1 value.

2- ***phase 2*** as shown in fig. 5 Only attack traffic is generated and destined to host 16.
In this case entropy value will be 0.



```
result = h1.cmd('sudo python Traffic.py -s 2 -e 17 ')

print 'ARP project', result
```

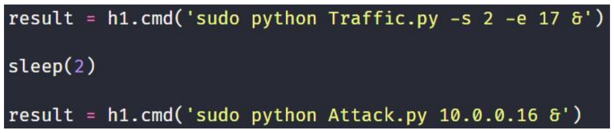
Figure 4. Phase 1 benign traffic generation command



```
result = h1.cmd('sudo python Traffic.py -s 2 -e 17 ')

print 'ARP project', result
```

Figure 5. Phase 2 Attack traffic generation command

3- ***Phase 3*** as shown in fig. 6 Both benign traffic and attack traffic are generated simultaneously via host 1. (IPMAC spoofing)
In this case, entropy value is around 0.5.



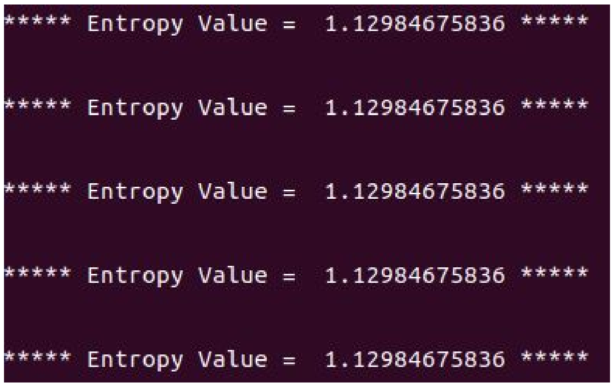
```
result = h1.cmd('sudo python Traffic.py -s 2 -e 17 &')

sleep(2)

result = h1.cmd('sudo python Attack.py 10.0.0.16 &')
```

Figure 6. Phase 2 Attack traffic generation command

For mitigation, we will limit the link capacity between the compromised host and the switch for a limited time. So the benign user will notice that and will start to do some countermeasures to protect his device against the attacker.
In our Algorithm, POX controller is using a new modified file from the original pox controller module so it can call the entropy-based algorithm. the modified file name is l3forwarding, in which we are importing our detection python file.
To monitor our network traffic, we used sFlow-RT, which is a monitoring module being used to watch real-time generated traffic, SDN topology and flow metrics [6].

V. RESULTS

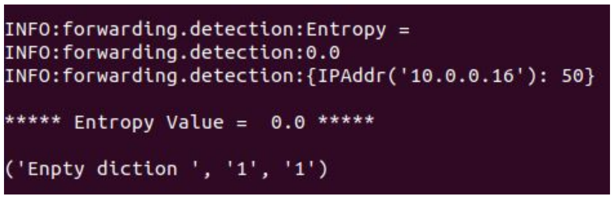
Firstly, we started our algorithm by running ***Phase 1*** and as depicted in Fig. 7 and the green area in Fig. 7



```
***** Entropy Value =  1.12984675836 *****


***** Entropy Value =  1.12984675836 *****


***** Entropy Value =  1.12984675836 *****


***** Entropy Value =  1.12984675836 *****


***** Entropy Value =  1.12984675836 *****
```

Figure 7. Phase 1 entropy value for 50 packets window size

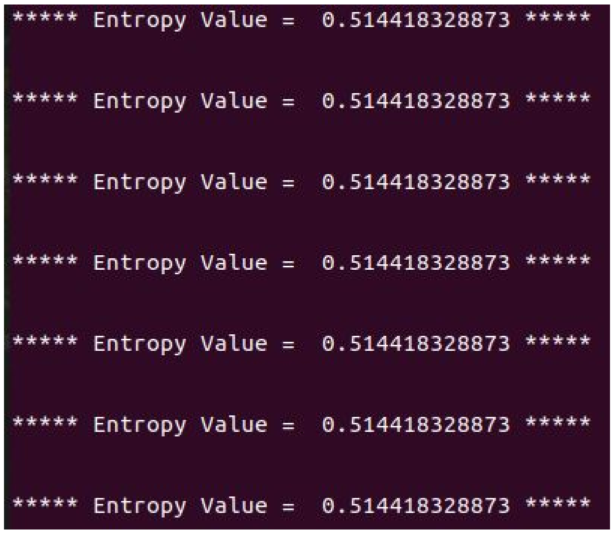
Secondly, we run ***Phase 2*** and as depicted in Fig. 5 and the red area in Fig. 8



```
INFO:forwarding.detection:Entropy =
INFO:forwarding.detection:0.0
INFO:forwarding.detection:{IPAddr('10.0.0.16'): 50}

***** Entropy Value =  0.0 *****

('Enpty diction ', '1', '1')
```

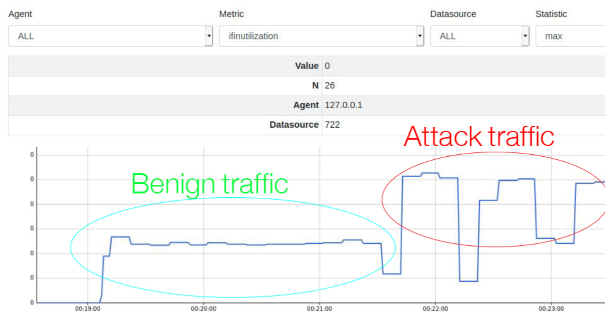
Figure 8. Phase 2 entropy value for 50 packets window size

Finally, we run ***Phase 3*** and as depicted in Fig. 9



```
***** Entropy Value =  0.514418328873 *****

***** Entropy Value =  0.514418328873 *****

***** Entropy Value =  0.514418328873 *****

***** Entropy Value =  0.514418328873 *****

***** Entropy Value =  0.514418328873 *****

***** Entropy Value =  0.514418328873 *****

***** Entropy Value =  0.514418328873 *****
```

Figure 9. Phase 3 entropy value for 50 packets window size



Agent: ALL
Metric: ifinutilization
Datasource: ALL
Statistic: max

Value	0
N	26
Agent	127.0.0.1
Datasource	722

Figure 10. sFlow-RT metric browser

To recap the algorithm, in phase 2 and phase 3 and after using the following flowchart depicted in fig. 11 we can detect that there is a DDos attack destined to a specific host.

As mentioned above, as a mitigation solution for a DDos detected attack, we can limit the link BW between the compromised host and switch.
This can be done via using a function in the main python file.

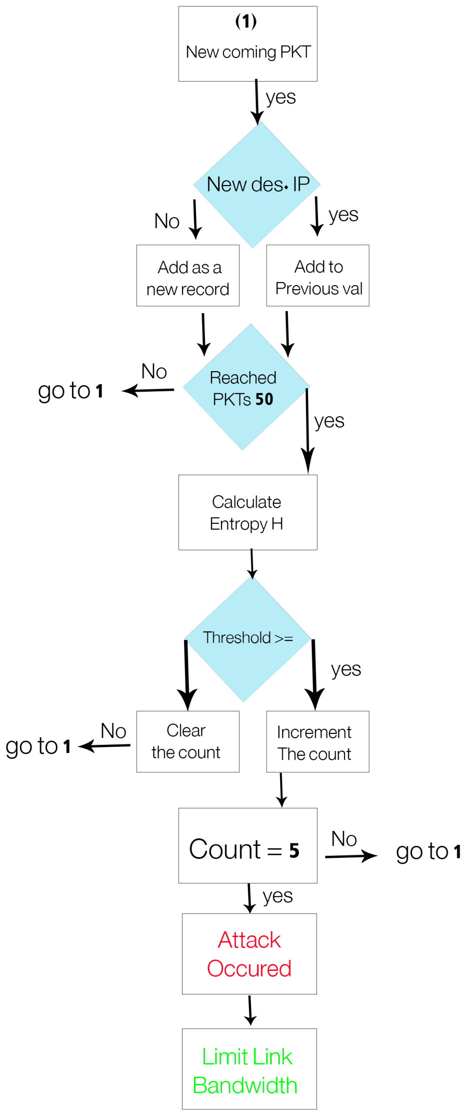


Figure 11. Algorithm flowchart

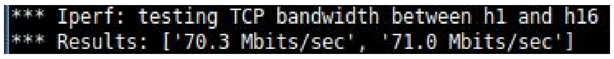
VI. Results

AS shown from above steps and based on the entropy-based detection method, we can detect DDos attacks in SDN and mitigate the attack.
The results show that as long as there is a five continuous window size traffic with an entropy value below calculated threshold, attack is detected and mitigation is implemented. It is also shown that if attacker launched discrete attack which means that there is no five continuous window size attack traffic, attack cannot be detected. In addition, number of generated packets and number of total hosts can affect the entropy value. Also, the window size and the threshold value need to be adjusted manually according to the number of hosts, total number of packets generated and the type of traffic. The

algorithm does not show noticeable overload on the CPU to detect the DDos attack (50 packets window size).
As a mitigation approach, we will use ***Iperf*** tool to measure the link bandwidth before and after the attack.
Iperf is multi-parameter cross-platform tool that can measure the loss, the bandwidth and many other parameters in IP networks [7]. Iperf support TCP,UDP and SCTP protocols. Using Iperf, we measured the link bandwidth between the source of attacking packets and the victim. And after using the following code,

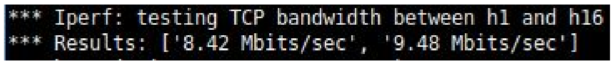
```
linkopts = dict(bw=10,
 delay='5ms',max_queue_size=100)
net.addLink(h1, h16, cls=TCLink,
 **linkopts)
```

you can decrease the link bandwidth to be 10Mb in fig. 13 instead of 100Mb as shown in fig. 12 and also we can change other parameters like the delay and Max queue size.



```
*** Iperf: testing TCP bandwidth between h1 and h16
*** Results: ['70.3 Mbits/sec', '71.0 Mbits/sec']
```

Figure 12. Iperf tool results before mitigation



```
*** Iperf: testing TCP bandwidth between h1 and h16
*** Results: ['8.42 Mbits/sec', '9.48 Mbits/sec']
```

Figure 13. Iperf tool results after mitigation

VII. Conclusion and future work

Entropy as a statistical method to detect destination IP randomness is an effective way.
Different challenges needs to be investigated in future work. Firstly, How can we block just the attacker traffic and allow the benign user traffic.
Secondly, How to differentiate between DDos attack and possible benign flash crowd traffic.
In future approaches it is recommended to use machine learning algorithms and try to compare between statistical methods and machine learning methods.
In addition, investigating false positive(FP), false negative (FN), True positive (TP) and True negative (TN) values.
Also, investigating various test cases based on different typologies, different traffic size and different hosts number.
Finally, investigating DDos attack in which the attacker is aiming to affect the controller availability via sending traffic to IP destinations that switch does not know so it will forward the request to the controller to consult it and this will overload the controller.

References

[1] URL: https://www.wired.com/story/github-ddos-memcached/#:~:text=11%3A01%20AM-,GitHub%20Survived%20the%20Biggest%20DDoS%20Attack%20Ever%20Recorded,platform%20GitHub%20all%20at%20once..

[2] URL: https://www.dw.com/en/malicious-attack-takes-wikipedia-offline-in-germany/a-50335521#:~:text=Users%20in%20Germany%20trying%20to,an%20error%20message%20and%20frustration.&text=The%20website%20fell%20victim%20to,in%20other%20parts%20of%20Europe..

[3] URL: https://www.edrawsoft.com/edraw-max/.

[4] URL: https://github.com/noxrepo/pox.

[5] URL: https://scapy.net/.

[6] URL: https://sflow-rt.com/.

[7] URL: https://iperf.fr/.

[8] Maryam Kia. “Early Detection and Mitigation of DDoS Attacks In Software Defined Networks”. In: (Feb. 2015).

[9] Fatima W. Hussein* Mahmood Z. Abdullah Nasir A. Al-awad. “Implementation of Entropy-based Distributed Denial of Service Attack Detection Method in Multiple POX Controllers”. In: (June 2019).

[10] Seyed Mohammad Mousavi. “Early Detection of DDoS in Software Defined Networks Controller”. In: (May 2014).

[11] Seyed Mohammad Mousavi and Marc St-Hilaire. “Early Detection of DDoS Attacks against SDN Controllers”. In: (June 2015).

[12] Narayan D G Nisharani Meti and V. P. Baligar. “Detection of Distributed Denial of Service Attacks using Machine Learning Algorithms in Software Defined Networks”. In: (2017).

[13] Cal-Poly Pomona. “Detection of DDoS in SDN Environment Using Entropy-based Detection”. In: (Nov. 2019).

[14] Jacir L. Bordim Ranyelson N. Carvalho and Eduardo A. P. Alchieri. “Entropy-based DoS Attack identification in SDN”. In: (May 2019).

[15] Bin Wu Runyu Li. “Early detection of DDoS based on $\phi - entropyinSDNnetworks$”. In: (June 2020).

[16] Mininet Team. URL: http://mininet.org/.