www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 6 Issue 12 December 2017, Page No. 22243-22249
Index Copernicus value (2015): 58.10, 76.25 (2016) DOI: 10.18535/ijecs/v6i12.01

Intelligent hybrid detection and mitigation of Distributed Denial of Service Attacks in SDN

Douglas Omuro Makori[1], Seckin Ari[2]

Department of computer engineering
Sakarya University Turkey

Associate Professor
Department of Computer Engineering
Sakarya University Turkey

Abstract
Distributed denial of service attacks are a becoming a norm in today's networked information systems. They were there, they are here and will continue being here because of the dynamic nature and difficulty in detecting the myriad types of DDoS attacks. They are even becoming more complicated by the coming of age of internet of things with so many devices being connected to the internet. Software defined networking is a new unexplored field so it might be given more attention by hackers because of the "cool" tag as a result of that propose a mechanism for detecting DDoS attacks in software networks through the use of machine learning support vector machines for detecting the DDoS attacks we also use entropy counter to be able to mitigate the DDoS attack by installing flow rules in Openflow switches to block the source of the flooding in the network.

Index Terms—Software defined networks, Distributed denial of service attacks, Support vector Machines, Network security

Introduction

Distributed denial of service attacks are one of the biggest nightmare scenarios that we have on the internet today. This problem is exacerbated by the emergence of internet of things field that is increasing the number of devices connected to the internet. Most of these devices are using default passwords which makes it easier for those devices to be taken control of and result in creation of botnets (Cobb, 2016). Some of the IoT devices that can be taken over include camera, teddy bears, Home Wi-Fi-routers, Baby monitors, DVRs among others that are connected to the internet. These devices are used to build large botnets that are used for launching massive DDoS attacks against big corporations. An example is the attack against Dyn an online optimization infrastructure company that affected services from Twitter, to Spotify and many other companies (Festin, 2016). What makes this attacks more worrying is that they are conducted by amateurs who use readymade scripts such as Mirai found on the internet to launch the DDoS Attacks such as the one against Dyn. SDN being a new field, it's gaining a lot of traction from the industry leading to its adoption in the enterprise. This in addition to prediction that it will half the cost of networks through softwarerization of the networks, hackers are giving it more priority. They are also giving it priority because of its "new" tag so that they look 'cool' or because of its high adoption in the enterprise, they will be able to access enterprise resources before well researched and reliable solutions are found. In this paper we propose a lightweight system that combines the use of machine earning support vector machines and entropy in detecting and mitigating DDoS attacks in SDN.

Software defined networks (SDN)

Software defined networking is a new networking phenomenon that is striving to change how networking is being done by transforming many network functions. This technology decouples the control and data plane which enables the network engineers to be able to program the networks to their needs. This is a move from the past when the control plane and the data plane were vertically integrated which led to delayed innovation in networks since the same software was used with legacy hardware until when hardware was retired or upgraded. This meant that until the hardware is changed, the software is the same. Software defined networks come in and redefine networking and bring in the concept of softwarerization and since software can evolve faster than hardware, SDN is promoting innovation in the networks especially the enterprise networks.

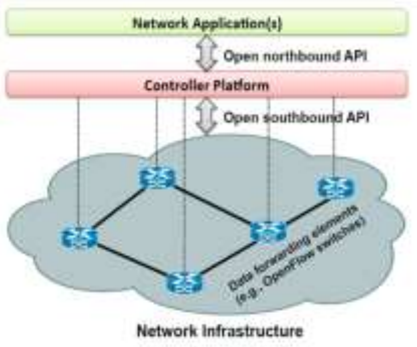
Imagine a network that can have a software firewall or load balancer; that's what software defined networks is enabling with softwarerization. SDN comes with three key abilities i.e. automation through programmability, network virtualization and separation of control and data functions. SDN promises increased network's utilization by allocating resources on demand, improving the network's reliability and flexibility, as well as making network management easier as well as facilitating the innovation of networks.

According to open networking foundation SDN is an emerging flexible architecture whose cost is cheaper, easier to manage and easier to adapt to the high demands of modern applications[1].The SDN architecture is unique in that its centrally managed, agile, programmable as well as based on open standards making it vendor neutral.

The SDN architecture is composed of the control plane and the data plane which are able to communicate with each other via the Openflow protocol/south-bound API that is vendor neutral

[1] https://www.opennetworking.org/sdn-resources/sdn-definition

and standard certified.



The concept of vendor neutral and open standards is a headache to network vendors such as cisco, Jupiter who consider their buyers as new customers since they need to buy new vendor specific add-ons such as firewalls and load balancers thus is an **expensive option.**

That definition is in tandem with Big switch's definition of SDN as a new approach to networking that does away with the complexities of the legacy networks and introduces dynamism through the use of standards based software abstraction between the forwarding plane as well as the control plane.

Data plane/Infrastructure plane

Infrastructure layer is where packet forwarding takes place. It is composed of physical switches or virtual switches (open virtual switch). It is able to communicate with the controller in the control plane through Openflow protocol (South-bound API).

Openflow protocol

This is the standardized communications protocol between the control plane/controller and the forwarding plane/switches. It is the southbound API for software defined networks. It enables network programmers to modify the behavior of the switches and routers through writing scripts that run in the controller. It does this through offering direct access and manipulation of forwarding plane devices such as switches and routers. Other than Openflow there are other SDN protocols like border gateway protocol for hybrid SDN, NETCONF which is responsible for configuring Openflow enabled devices, MPLS-TP a transport profile for multiprotocol label switching used as a network layer technology for transport networks, Open vSwitch Database Management Protocol (OVSDB) an Openflow configuration protocol for managing open vSwitch implementations in SDN and lastly, Extensible Messaging and Presence Protocol (XMPP) for messaging and online presence detection all in real time.

Openflow Messages

They facilitate communication between the controller and switches. Openflow messages also lead to particular events being invoked. Some of the messages include:

Feature request

This message is sent by the controller to the switch requesting for switch features which will enable the controller know the switchs capabilities. It's composed of just the Openflow header and feature request value already set.

Features reply

This is the reply of the switch to the ofpt_features_request from controller. It normally enumerates the capabilities of the switch.

PacketOut

The *ofp_packet_out* is sent from the controller to the switch instructing the switch to send a packet or enqueue it or even discard it. Its attributes include Buffer_id, in_port, actions and the data (bytes) to be sent. The controller uses OFPT_PACKET_OUT to initiate PacketOut.

FlowRemoved

The FlowRemoved message is used to notify the controller that flow has timed out.

It has the following fields, Openflow header, Openflow_match, cookie field, and packet count and byte count fields. It also has duration in both nanoseconds and seconds. This message also includes reason field which explains why the flow was removed e.g. due to idle_timeout or hard timeout. The switch uses OFPT_FLOW_REMOVED to initiate the FlowRemoved message.

FlowMod

This is a message with instructions to modifying the flow table. Important fields in flow modification message are hard_timeout, idle_timeout in that they determine how fast flows expire. These fields can be used write flows or delete flows that are suspect. OFPT_FLOW_MOD is used by the controller to initiate FlowMod message.

Statistics messages

a) Individual Flow Statistics
Individual flow stats are requested by OFPST_FLOW.
b) Aggregate Flow Statistics
This is statistics about multiple flows in Openflow. It's requested by OFPST_AGGREGATE.
c) Table Statistics
Table information is requested by OFPST_TABLE.
d) Port Statistics
Physical statistics is obtained by OFPST_PORT request type.

PacketIn Message

This message is from the switch to the controller. It's raised when packets arrive in the datapath or switch and don't match all fields thus sent to the controller to determine what to do with them. OFPT_PACKET_IN is used by the controller for packet-in message. Its key attributes are Port (int) of the in port and Data (bytes) which is normally raw bytes.

Packetin is also important especially for DDoS de-tection especially the packet spoofing DDoS attacks in that you can monitor the packet-in packet-out and flow removed to decide accurately if it's a DDoS attack.

Controller

This is considered the brain of the SDN network. It manages the flow control to the switches and routers below via southbound APIs which most of the time is Openflow protocol which enables the communication between the switches and the controller. It also controls business applications and logic via northbound APIS.

The controller supports north-bound APIs which enables developing of applications that extend the ability of the SDN controller.

These are the two most common techniques that are and can be used to take over the controller:

- Flow decision requests: this happens when a controller is flooded with flow decision requests until the controller's computing resources are overwhelmed thus goes offline rendering the rest of the network useless especially when the hard timeout elapses.
- Flow entry flooding: this is also possible when falsely crafted flows are directed at openflow switches and their hard timeout is set to be long rendering the switches full and cannot be able to do any forwarding

decisions.

Controller	Programming language	Features
POX	Python	Good documentation Good for research and simulation
RYU	Python	REST API Good documentation
Floodlight	Java	REST API Excellent documentation
Opendaylight	Java	Rest API Excellent documentation Production ready Steep learning curve
NOX	C++/Python	Excellent for where performance is critical Steep learning curve
Beacon	Java	Rest API

Figure 1 Most popular SDN Controllers

Distributed denial of service attacks

According to Akamai a leading content delivery network, DDoS is an attempt to make online services unavailable or making them to respond slowly[2]. Despite being the oldest form of network attacks, they keep changing in volume and sophistication according to Akamai. According to Kaspersky labs instances of DDoS attacks have increased as a result of increase in cheap DOS tools which has resulted in the increase in the DDoS attacks[3]. It also says that there has been an increase in the complexity of DDoS attacks e.g. application level and https attacks. It highlights an instance of a combined attack on the Moscow stock exchange where (SYN + TCP Connect + HTTP-flood + UDP flood) were used.

DDoS attacks may fall into these categories:

[2] https://www.akamai.com/us/en/resources/ddos-attacks.jsp

[3] https://securelist.com/ddos-attacks-in-q1-2017/78285/

Protocol attacks

These ones target server resources and the peripheral devices such as load balancers or firewalls. It includes Ping of Death, SYN floods, Smurf DDoS, fragmented packet attacks and others it's normally measured as packets per second.

Application layer attacks

These include zero day attacks that target operating system vulnerabilities. It can also include GET/POST floods as well as those attacks targeting Apache. It is intended at crashing the server. It normally attacks a server or an application. It's difficult to detect because it comprised of both genuine and malicious traffic.

Volume based attacks

These ones include ICMP flood, UDP flood, TCP flood and any other spoofed Packet attacks with the intention to saturate target's bandwidth. It's normally measured in bits per second.

ICMP attacks

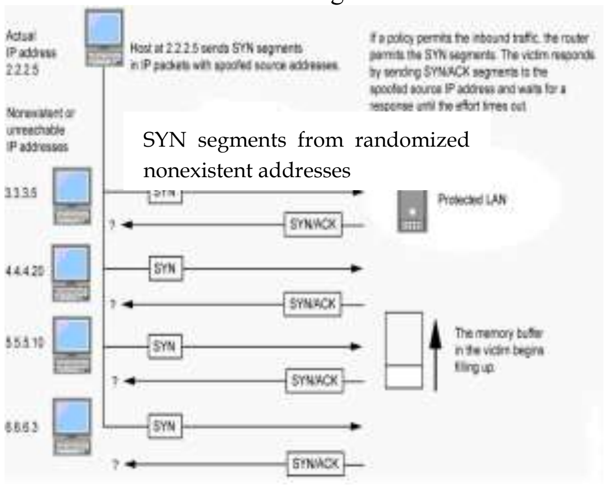
This attacks generally overwhelms the receiver with echo request /ping packets. It normally achieves that by sending as many packets as possible without necessarily waiting for replies.

UDP Flooding

This attack takes advantage of the sessionless user datagram protocol (UDP) to flood UDP packets to random ports on a remote host which will force it to check on which particular application is listening on that particular port leading to destination unreachable every time which eats the system resources which might make the host to go offline.

Sync flood

This DDoS attack targets the three way handshake of TCP protocol where a host receives a synchronized message to begin the handshake and the server acknowledges the message and sends an acknowledgement flag to that host which in turn closes the connection. However, spoofed messages can be sent which will make the connection not to be close and as a result packets will be continuously generated until the server is overwhelmed as demonstrated in the figure below.


Actual IP address 2.2.2.5
Host at 2.2.2.5 sends SYN segments in IP packets with spoofed source addresses.
If a policy permits the inbound traffic, the router permits the SYN segments. The victim responds by sending SYN/ACK segments to the spoofed source IP address and waits for a response until the effort times out
Nonexistent or unreachable IP addresses
SYN segments from randomized nonexistent addresses
3.3.3.5
4.4.4.20
5.5.5.10
6.6.6.3
SYN
SYN/ACK
?
Protected LAN
The memory buffer in the victim begins filling up.

Common DDoS Attacks in SDN

Protocol based attacks	Volume based attacks
ICMP	Sync
UDP	Smurf

Review of related works

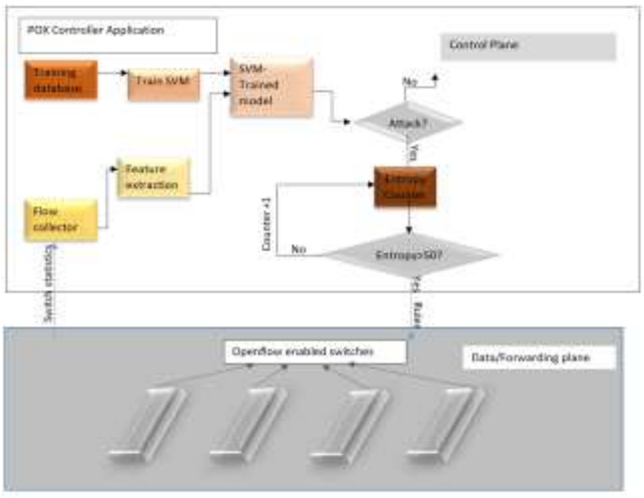
Software defined networking faces a myriad of security problems (Collings and Liu, 2014; Sezer et al., 2013; Yan and Yu,

2015). Some of the main security issues about SDN are discussed by (Kreutz et al. 2013). Some of the vulnerabilities in SDN include single point of failure with the controller as well as the issue of trust between the controller and the other devices so it is very easy for an attacker to masquerade and attack the network from within. However, the softwarerization of networks in SDN is enabling easy network monitoring, network forensics, network programming as well as implementation of flexible policies (Sezer et al., 2013).
Among the biggest security challenges in software defined networks is distributed denial of service attacks and as a result of that it has been researched widely. DDoS attack detection has been deeply researched by (Braga et al., 2010; Mehdi et al., 2011; Miao et al., Wang et al., 2015; Li et al., 2014) while DDoS mitigation has been discussed by Wang et al., 2015; Kampanakis et al., 2014, Giotis et al., 2014).

DDoS detection in legacy networks
Distributed denial of service attacks have been causing problems with legacy networks and as a result a lot of research as gone into detecting them. Swati & Gupta(2012) used genetic algorithm with the KDD dataset to detect intrusion detection such as DOS,R2L,U2R.Jeya et al.(2012) use an IDS which uses Fisher Linear Discriminant Analysis and the KDD intrusion detection dataset to able to detect intrusions. It detects DOS, R2L, U2R attacks efficiently

DDoS Detection in SDN
Mehdi et al. (2011) uses four anomaly detection algorithms i.e. rate limiting, entropy detector calculator, TRW-CB and NETAD to differentiate normal from abnormal traffic. Shinet et al. (2013) propose AVANT-Guard which is a mechanism of extending the Openflow enabled switch in order to detect anomalies. Wang et al. (n.d) propose OF-GUARD which is a mechanism that works by migrating table miss packets and using proactive flow rules to distinguish fake packets. Braga et al. propose a lightweight DDoS detection tool that is based on flow information. This is perceived to have low overhead compared to other detection systems such as deep packet inspection which uses the packet header information as well as the packet payload information. Kim et al. propose a flow based method for detecting abnormal traffic, this system's flow generator had its flows stored in a database something that will increase overhead costs for the network. Phan et al. (2016) propose a hybrid system that utilizes support vector machines and self-organizing maps that act on flow statistics where features are extracted and the traffic classified by the use of support vector machines and self-organizing maps which has been trained by the use of a training database. The result is a flow rules which are implemented as policies.

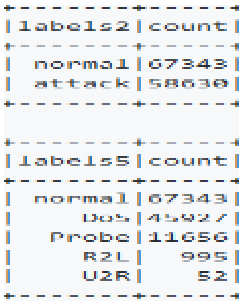
Proposed hybrid DDoS detection system using SVMs and Entropy



Dataset /Training database
In this paper we use machine learning supervised classification support vector machines algorithm which we train using the NSL-KDD dataset from the Canadian institute of cyber security. In addition to this, because of too many false positives involved with using machines in intrusion detection (Landress, A.2016) we do feature selection so that we use the most important features in order to reduce the false positives. Entropy is also used to monitor the packets and bytes in each flow for any slight change. This is used to complement the machine learning module that has a trained SVM model.
According to Tavallaee et al (2009) NSL-KDD dataset has the following advantages over the old KDD99 cup;

A. Removal of redundant records in the training datasets which reduces biased results towards the frequency of records.
B. Reduction in the amount of records in the testing and training datasets which enabled the use of all records instead of portions of the record. This has enabled comparison of the result from different intrusion detection researches using data.
C. Test datasets is duplicate free thus performance of the learning models is not influenced by the frequency of records.
D. Availability of testing and training datasets in various formats i.e. text files, Weka files (arff) and PCAP formats which makes it easier to use in different scenario.

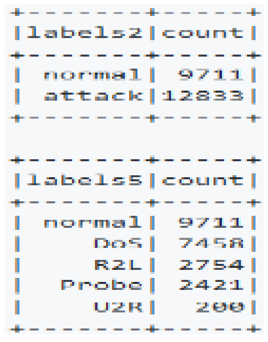
Support vectors for intrusion detection
In the proposed system a support vector machine is trained using the NSL_KDD training dataset. The SVM has a C kernel as a parameter because it will enable it to have more freedom to select more samples as support vectors. We checked the descriptive statistics of the dataset features is shown below:



```
+-------+-----+
|labels2|count|
+-------+-----+
| normal|67343|
| attack|58630|
+-------+-----+

+-------+-----+
|labels5|count|
+-------+-----+
| normal|67343|
|    DoS|45927|
|  Probe|11656|
|    R2L|  995|
|    U2R|   52|
+-------+-----+
```

Figure 2 training dataset label count



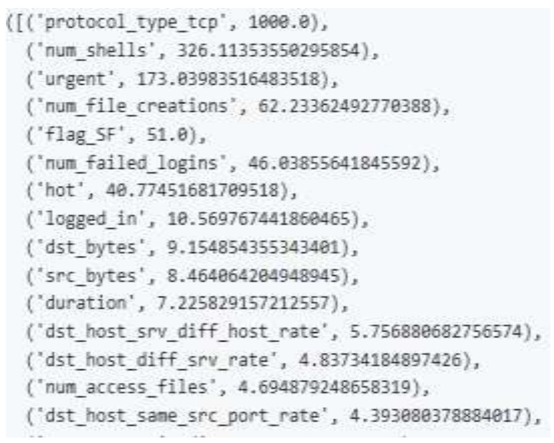
```
+--------+------+
|labels2|count|
+--------+------+
|  normal| 9711|
|  attack|12833|
+--------+------+

+--------+------+
|labels5|count|
+--------+------+
|  normal| 9711|
|     DoS| 7458|
|     R2L| 2754|
|   Probe| 2421|
|     U2R|  200|
+--------+------+
```

Figure 3 Testing Dataset label count

Feature selection

In order to avoid inconsistent results that come as a result of using too many features as found out by (Varma et al, 2016) we used attribute ratio as proposed by (Chae & Choi, 2014) for feature selection. The features are listed from the most important in descending order in the table below.



```
([('protocol_type_tcp', 1000.0),
  ('num_shells', 326.1135355029585 4),
  ('urgent', 173.03983516483518),
  ('num_file_creations', 62.23362492770388),
  ('flag_SF', 51.0),
  ('num_failed_logins', 46.03855641845592),
  ('hot', 40.77451681709518),
  ('logged_in', 10.569767441860465),
  ('dst_bytes', 9.154854355343401),
  ('src_bytes', 8.464064204948945),
  ('duration', 7.225829157212557),
  ('dst_host_srv_diff_host_rate', 5.756880682756574),
  ('dst_host_diff_srv_rate', 4.83734184897426),
  ('num_access_files', 4.694879248658319),
  ('dst_host_same_src_port_rate', 4.393080378884017),
```

Figure 4 Most significant features in the training Dataset

Other techniques have been used to select features such as info gain ratio (Oreski &Novosel, 2014), gain ratio (Modinat et al 2015). This techniques are very important because they help in weeding out features that are insignificant and if they are included in the training these features are likely to create a bias towards frequency. This will make the model inaccurate in predicting the final results.

Flow logger/Flow collector

This is a module that asks for flow entry information from Openflow switches' tables periodically through the use of Openflow stats_request to the switches which reply to the controller with ofp_stats_reply.

Feature extractor

This module extracts important features from the flows that are key for detection of DDoS attacks in software define networks. The most important fields extracted are source and destination IP address and port, protocol type are extracted from the headers while the received and sent Packets and bytes as well as the duration(in seconds and nanoseconds) are extracted from the counters. All these features are collected on a per flow basis. These fields closely match the most important features selected by attribute ratio.

SVM classifier

This module is composed of a trained SVM model that has been trained via extracted data from NSL-KDD dataset. The decision to extract the features instead of using all features was to help avoid the bias that normally favors frequency. It also helps to decrease false positives. The classifier module compares features from the feature extractor them with the trained model to determine which flow correspond to a DDoS attack.

Entropy counter

Because of the many cases of false positives coming into play when using machine learning for intrusion detection especially when the model is not tuned enough, we use an entropy counter that further checks if really it is a DDoS attack or a false positive. In our particular experiment we use average packets per second per flow. Sometimes there are problems with counting packets per flow per second since sometimes initially in the connection the number of packets can spike before they normalize, this increases chances for false positives. We put a ceiling of 50 packets per second which if reached that particular ip address is blocked only on condition if that particular IP address's flow was classified as an attack by tha classifier. As a result of this the attack can be detected and mitigated. It is noticed that during that attack, the number of packets arrived increased linearly in a sharp manner. This is normally as a result of the packets that are not arriving at the switch not matching as a result are packaged as packet in and sent to the controller which is to determine the next course of action.

Experiment

In our testbed pox controller is used with Openflow 1.0. Mininet is used as the network emulator and python is used as a programming language of choice as POX is python based. For flood packet generation, hping3 is used for generating a flood of packets from spoofed IP addresses.

Discussion and Results

In order to find the accuracy of our model performance on a real world dataset, we use the following measures to determine the performance of our model.

Accuracy- The classification accuracy of the model is the accurate number of correct predictions from the total predictions. We fine-tuned the model further to ensure more accuracy thus was suitable to solve our problem. Other than accuracy, precision, recall and even F1 score can be used.

	Positive	Negative
Positive	True Positive(TP)	False Positive(FP)
Negative	False Negative(FN)	False Negative(FN)

Figure 5 Confusion Matrix

Precision- This is the ratio of correctly predicted positive predictions

$$precision(p) = \frac{TP}{TP + FP}$$

Recall- This is the count of all the positive occurrences regardless of if they were correctly predicted by the model that is being used. It's also called sensitivity.

$$Recall(R) = \frac{FP}{TP + FN}$$

F1 Score- This is the weighted average of precision and recall

$$F1\ Score = 2 * \left(\frac{Precision * Recall}{Precision + Recall}\right)$$

Recall and precision work well on an unevenly distributed dataset and normally concentrate on positives rather than negatives thus it is normally necessary to assign "positive" base to the values of interest.
Depending on class distribution and the cost of FN and FP, the following measures should normally be given more emphasis as show in the table below

		Class distribution	
		EVEN	UNEVEN
Cost	FN cost More	Recall	Recall
	Same cost for FN &FP	Accuracy/F1 score	F1 score
	FP cost more	Precision	Precision

Figure 6 recommendation on when to use various measures

Precision and recall can easily be influenced by variation of the threshold of the model. The optimum threshold can be calculated basing on cost of false positives and false negatives (Ruiter, A. 2015).
Threshold is also very important when fine tuning a model as it is able to greatly reduce the amount of false positives as show in

	normal	DoS
normal	13327	1
probe	3	9189

Accuracy = 0.999822

AUC = 0.999799

False Alarm Rate = 7.503e-05

Detection Rate = 0.999674

F1 score = 0.999782

	precision	recall	F1-score	support
0.0	1.00	1.00	1.00	13328
1.0	1.00	1.00	1.00	9192
Avg/total	1.00	1.00	1.00	22520

Figure 7 Detection with Threshold not optimised

	normal	attack
normal	13248	80
attack	427	11378

Accuracy = 0.979827

AUC = 0.978913

False Alarm Rate = 0.0060024

Detection Rate = 0.963829

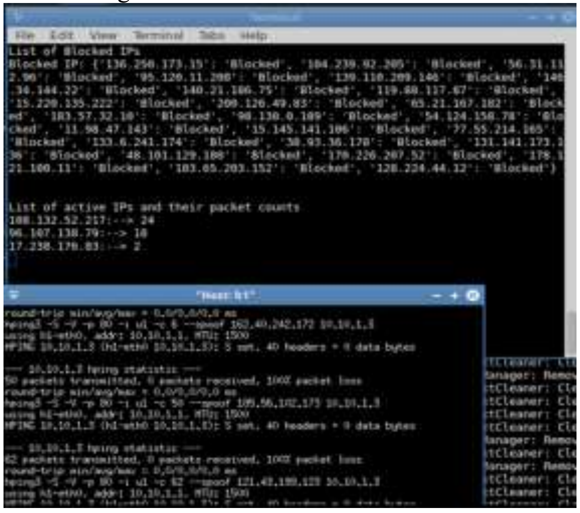
F1 score = 0.978206

	precision	recall	F1 score	Support
0.0	0.97	0.99	0.98	13328
1.0	0.99	0.96	0.98	11805
Avg/Total	0.98	0.98	0.98	25133

Figure 8 Detection with optimized Threshold

From the above figures we can be able to deduce that threshold helps to increase detection rate of algorithms.

Discussion
In our DDoS detection system, we extract the most relevant features from the NSL-KDD dataset and use it to train a machine learning model which accurately detects a DDoS attack. This model can accurately detect a DDoS attack on real time traffic with an accuracy of 96.4%. In addition, we use an entropy of the count of packets per second if the flow is classified to be an attack and in this case a flow rule is written to the switches to block that IP that is flooding.



```
List of Blocked IPs
Blocked IP: {'136.250.173.15': 'Blocked', '104.239.92.205': 'Blocked', '56.31.11
2.90': 'Blocked', '95.120.11.200': 'Blocked', '139.110.209.146': 'Blocked', '140
.30.144.22': 'Blocked', '140.21.186.75': 'Blocked', '119.60.117.67': 'Blocked',
'15.220.135.222': 'Blocked', '209.126.49.83': 'Blocked', '65.21.167.182': 'Block
ed', '183.57.32.10': 'Blocked', '98.130.0.189': 'Blocked', '54.124.150.78': 'Blo
cked', '11.98.47.143': 'Blocked', '15.145.141.106': 'Blocked', '77.55.214.165':
'Blocked', '133.6.241.174': 'Blocked', '38.93.36.170': 'Blocked', '131.141.173.1
36': 'Blocked', '48.101.129.188': 'Blocked', '170.226.207.52': 'Blocked', '178.1
21.100.11': 'Blocked', '183.65.203.152': 'Blocked', '128.224.44.12': 'Blocked'}

List of active IPs and their packet counts
108.132.52.217:--> 24
96.107.138.79:--> 10
17.238.170.83:--> 2
```

```
round-trip min/avg/max = 0.0/0.0/0.0 ms
hping3 -S -V -p 80 -i u1 -c 6 --spoof 162.40.242.172 10.10.1.3
using h1-eth0, addr: 10.10.1.1, MTU: 1500
HPING 10.10.1.3 (h1-eth0 10.10.1.3): S set, 40 headers + 0 data bytes

--- 10.10.1.3 hping statistic ---
50 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
hping3 -S -V -p 80 -i u1 -c 50 --spoof 105.56.102.175 10.10.1.3
using h1-eth0, addr: 10.10.1.1, MTU: 1500
HPING 10.10.1.3 (h1-eth0 10.10.1.3): S set, 40 headers + 0 data bytes

--- 10.10.1.3 hping statistic ---
62 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms
hping3 -S -V -p 80 -i u1 -c 62 --spoof 121.43.199.123 10.10.1.3
using h1-eth0, addr: 10.10.1.1, MTU: 1500
```

Figure 9 Flooding by hping3/Blocked flooding IP addresses

We were able to to detect flooding IPs and blocked them in real-time thus we were able to protect the controller from being overwhelmed. Despite being an accurate, in future will we will look at implementing a deep packet inspection with machine learning solution which should be abit lightweight unlike normal DPI solutions. It should also be more accurate compared to our current solution.

Conclusion
This system used a simple support vector machine with C kernel trained with features extracted from NSL-KDD dataset and it was able to accurately detect DoS Attacks in a real-time software defined network. This enabled the entropy counter to be able to mitigate the DDoS attack with more accuracy. In future we plan to use a statistical entropy derived from the payload of the packet. When we use this with a machine learning algorithm it is going to be even more accurate.

References

1. Collings, J., Liu, J., 2014. An openflow-based prototype of sdn-oriented stateful hardware firewalls. In: Proceedings of ICNP, Raleigh, USA, IEEE, pp. 525–528.
2. Sezer et al (2013) are we ready for sdn? Implementation challenges for software-defined networks. IEEE Commun. Mag. 51 (July (7)), 36–43.
3. Chae & Choi(2014) Feature Selection for efficient Intrusion Detection using Attribute Ratio, International journal of computers and communications, pp. 22-28
4. Shin, S., Gu, G., 2013. Attacking software defined networks: a first feasibility study. In: Proceedings of HotSDN, Hong Kong, China, ACM, pp. 165–166.
5. Shin, S., Yegneswaran, V., Porras, P., Gu, G., 2013. Avant-guard: scalable and vigilant switch flow management in software-defined networks. In: Proceedings of CCS, Berlin, Germany, ACM, pp. 413–424.
6. Modinat et al (2015), Gain Ratio and Decision Tree Classifier for Intrusion Detection. *International Journal of Computer Applications,* nd, pp. 34-38
7. Oreski &Novosel, (2014), Comparison of Feature Selection Techniques in Knowledge Discovery Process. *TEM Journal,* UIKTEN
8. Tavallaee et al (2009) “A Detailed Analysis of the KDD CUP 99 Data Set,” Submitted to Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA), 2009.
9. Jeya et al.(2012) Efficient Classifier for R2L and U2R Attacks. In International Journal of Computer Applications Volume 45– No.21
10. Kohonen, T, “The self-organizing map,” Proceedings of the IEEE,vol. 78, no. 9, pp. 1464–1480, 1990.
11. M. Kim, H. Kang, S. Hung, S. Chung, and J. Hong, “A flow-based method for abnormal network traffic detection,” in IEEE/IFIP Network Operations and Management Symposium, 2004, pp. 599–612.
12. Wang et al.(2012) OF-GUARD: A DoS Attack Prevention Extension in Software-Defined Networks. Texas A&M University
13. Landress, A.(2016) "A hybrid approach to reducing the false positive rate in unsupervised machine learning intrusion detection," *SoutheastCon*, Norfolk, VA, 2016, pp. 1-6.
14. Varma et al(2016) Feature selection using relative fuzzy enthropy and ant colony optimization applied to real-time intrusion detection system. International conference on computational modelling and security. Bengaluru, 2016, Elsevier,B.V
15. Software-defined networking: the new norm for networks. Website. ⟨https://www.opennetworking.org/⟩.
16. https://www.opennetworking.org/sdn-resources/sdn-definition
17. https://openflow.stanford.edu/display/ONL/POX+Wiki
18. http://sdnhub.org/tutorials/pox/
19. Festin, R. (2016, 10 27). Who Was Responsible For Dyn DDoS Attack Last Week? Flashpoint Blames Amateur Hackers. Retrieved from TEchTimes:http://www.techtimes.comarticles/183797/20161027/who-was-responsible-for-dyn-ddos-attack-last-week-flashpoint-blames-amateur-hackers.htm
20. Cobb, S. (2016, 10 24). *10 things to know about the October 21 IoT DDoS attacks*. Retrieved from https://www.welivesecurity.com/2016/10/24/10-things-know-october-21-iot-ddos-attacks/