Düzce University
Journal of Science & Technology

Research Article

Anomaly Detection in Software-Defined Networking Using Machine Learning

Soumaine BOUBA MAHAMAT [a,*], Celal ÇEKEN [b]

[a] *Department of Computer and Information Engineering, Sakarya University, Sakarya, TURKEY*
[b] *Department of Computer and Information Engineering, Sakarya University, Sakarya, TURKEY*
* *Corresponding author's e-mail address: soumaine.mahamat@ogr.sakarya.edu.tr*

ABSTRACT

In recent years, the Software-Defined Networking (SDN) approach has emerged that aims to make computer networks more flexible. Although the SDN application on Google's internal network demonstrates the usefulness of the Software-Defined Network approach and the promise of future technology, security is a vital concern that cannot be ignored. In the SDN architecture, the attacker can now attack the network from any of the three planes because the Data Plane is separated from the Control Plane. Machine learning algorithms are methods used to detect attacks and intrusions on computer networks and can also be used for SDN. In this study, a new testbed has been implemented for anomaly detection using machine learning algorithms in SDN. The developed system analyzes flows passing through the OpenFlow supported switch and tries to detect abnormal situations using the decision tree machine learning algorithm. The results show that the system constructed using the decision tree algorithm works successfully against Distributed Denial of Service (DDoS) attacks.

Keywords: *Software Defined Networking, Anomaly Detection, Machine Learning, POX Controller*

Yazılım Tanımlı Ağlarda Makine Öğrenimi ile Anomali Tespiti

ÖZET

Son yıllarda, bilgisayar ağlarını daha esnek bir hale getirmeyi amaçlayan Yazılım Tanımlı Ağ yaklaşımı ortaya çıkmıştır. Google'ın iç ağındaki Yazılım tanımlı ağ uygulaması, Yazılım Tanımlı Ağ yaklaşımının kullanışlılığını ve gelecek vadeden bir teknoloji olacağını kanıtlamasına rağmen güvenlik konusu göz ardı edilemeyecek hayati bir sorundur. SDN mimarisinde, Veri Düzlemini Kontrol Düzleminden ayrıldığı için saldırganlar artık üç düzlemden herhangi birinden ağa saldırabilirler. Makine öğrenimi algoritmaları, bilgisayar ağlarına yapılan saldırıları ve izinsiz girişleri tespit etmede kullanılan yöntemlerdir ve Yazılım Tanımlı Ağlar için de kullanılabilir. Bu çalışmada, Yazılım Tanımlı Ağlarda makine öğrenme algoritmaları kullanılarak anomali tespiti için yeni bir test düzeneği geliştirilmiştir. Oluşturulan sistem OpenFlow destekli anahtar cihazından geçen akışları inceler ve karar ağacı makine öğrenmesi algoritmasını kullanarak anormal durumları tespit etmeye çalışır. Elde edilen sonuçlar karar ağacı algoritması kullanılarak oluşturulan sistemin DDoS saldırılarına karşı başarılı bir şekilde çalıştığını göstermiştir.

Anahtar Kelimeler: *Yazılım Tanımlı Ağ, Anomali Tespiti, Makine Öğrenimi, POX*

Geliş: 13/06/2018, Düzeltme: 14/08/2018, Kabul: 18/09/2018

I. INTRODUCTION

The networking world is trying to catch up with the other branches of IT such as the Server Virtualization world. Networking folks are working hard to make networks more programmable. Computer networks are difficult to manage. They are made up of a lot of devices ranging from routers and switches to middleboxes. Unfortunately, nowadays, these devices are being managed on a box-by-box basis and network outages are mostly the direct results of human errors in configuring the networking devices [1-5]. Moreover, there is no platform that would allow us to have a single view of the network and can help us manage all of its components. These are among the reasons why Software Defined Networking (SDN) is happening. SDN is a promising technology offering many possibilities. Vendors such as Cisco, HP, Juniper, etc. are greatly benefiting from the innovations that SDN is offering. In fact, Cisco has even revised its Certification track to include SDN related topics [6].

SDN is made up of three planes or layers (Figure 1): The Infrastructure Layer (Data Plane), The Control Layer (Control Plane), and the Application Layer (Application Plane) [7]. The Data Plane is where OpenFlow switches reside. End devices are directly connected to the switches as this plane. The Control Plane is the brain of SDN. SDN Controllers are found at the control plane and as their name suggest, they control the network. Application Developers can develop applications that will reside at the application plane. These applications communicate with the control plane through Application Programming Interfaces (APIs).

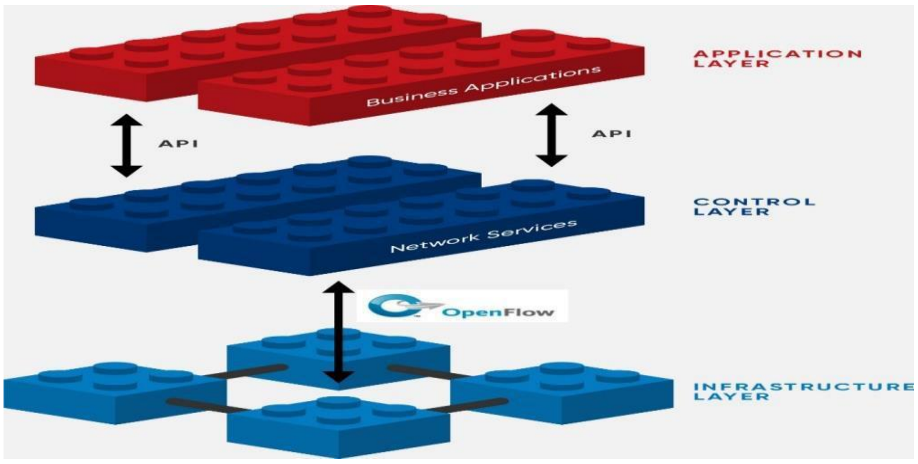


***Figure 1**. SDN Architecture [7]*

This decoupling makes SDN networks easier to manage and provides a platform for innovation. With the blessings of SDN comes its curse. Decoupling the Control Plane from the Data Plane has its advantages; however, it opens up the doors to attacks. SDN networks can be attacked in many different ways. The control plane is considered to be the brain of SDN and presents the risk of a single point of failure. If an attacker manages to compromise the controller, then the whole network is compromised. Despite this fact, [8] mentions that the benefits of a centralized control plane far outweigh the risk of single point of failure. The single point of failure issue can be alleviated by ensuring high availability systems. To tackle this and other issues, security risks related to SDN could be mitigated through encryption, deep packet inspection, access control lists, etc. [9].

Machine learning techniques have long existed and used in different areas of computer science. With the emergence of SDN, researchers thought about implementing machine learning algorithms in this new technology. In a research paper titled knowledge-defined networking [10], the authors described experiments whereby data is gathered from hosts, processed using machine learning, and used the results to manage the SDN controller configuration and forwarding. A joint effort between Cisco and Ericsson resulted in a framework named MILA (Machine Learning and Artificial Intelligence Framework). Using time series data in OpenDaylight, they presented a demo on how to apply machine learning in SDN and generate meaningful data from it [11].

Different attempts have been done to detect anomalies in SDN using Machine Learning. One of the most popular examples is a paper written by Braga et al. [12] in which they proposed a DDoS flooding attack detection mechanism using the NOX controller and the OpenFlow protocol. However, NOX is a deprecated controller and is no longer used nowadays. Figure 2 demonstrates the detection loop they used in their implementation. As shown in Figure 2, the IDS was developed as a NOX application. There system is made up of a flow collector, a feature extractor, and a classifier. The flow collector module collects flows from the OpenFlow switches. These flows are then passed on to the feature extractor module, which extracts six features. These features are: Average of Packets per flow (APf), Average of Bytes per flow (ABf), Average of Duration per flow (Adf), Percentage of Pair-flows (PPf), Growth of Single flows (GSf), and Growth of Different Ports (GDP). These features are then given as an input to the Self Organizing Maps (SOM) algorithm to do the classification. After classification, if a flow was classified as an attack, then an alert will be triggered.

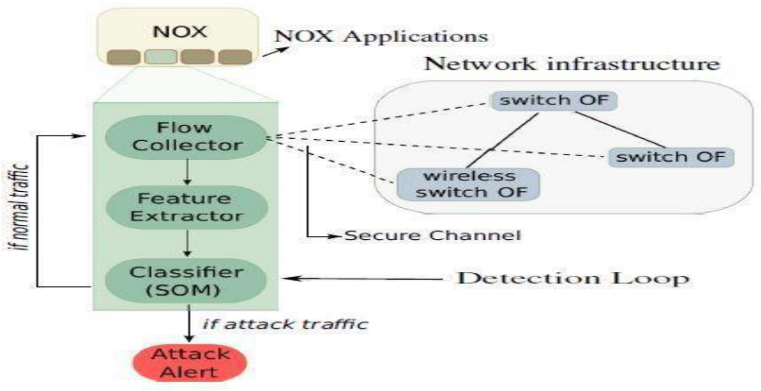


Figure 2. *Detection Loop Operation [12]*

Abubakar and Pranggono proposed a machine learning based intrusion detection system in SDN. However, they haven’t implemented the algorithm into the SDN structure and left it as future work. Dongsoo, Lee wrote his thesis on Improving Detection Capability of flow-based IDS in SDN [13]. In fact, part of our work is based on his thesis. Nonetheless, the author didn’t provide the machine learning algorithm, thus we ended up building our own algorithm to detect DDoS attacks in the network. Moreover, instead of using the KDD99 dataset, we used flow statistics from the switches in our network and made our own dataset out of them. We then trained our algorithm with that data and used it to do the anomaly detection.

In this research, we propose an anomaly detection strategy in SDN using machine learning. Using the POX controller, we developed an IDS module to detect attacks and block them. Features are extracted from flow statistics in the flow tables and sent to the flow-based detection module. The Scikit-learn based machine learning algorithm takes as input the features extracted and classifies the flow as either benign or malicious. Our contributions are summarized as follows:

- Usage of our internal network’s data as a dataset instead of using outdated datasets such as the KDD-99.
- Flexibility in changing the machine learning algorithm with a single line of code.
- We can easily update our dataset to cover new attacks.

The rest of the paper is organized as following: In Section II, we dive into the topic of intrusion detection in SDN using Machine Learning. Section III discusses the implementation of our testbed and the results. Finally, we will conclude the paper and discuss future work in Section IV.

II. Intrusion Detection In SDN Using Machine Learning

A. ANOMALY-BASED INTRUSION DETECTION

In computer networking, intrusion detection is a technology that allows us to detect suspicious activities and trigger an alarm when such activities are detected. Intrusion detection systems can be classified in terms of location or detection methods as shown in Figure 3. Intrusion detection systems in general have the same basic goal: detect intrusions and alert the network administrator in some way. If they are able to prevent the detected attacks, then they are called Intrusion Detection and Prevention Systems (IDPS).

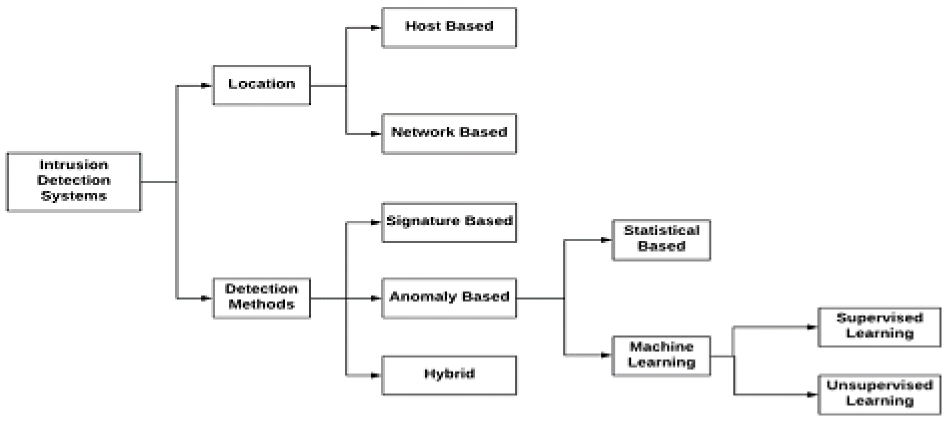


Figure 3. *Intrusion Detection Systems: A Taxonomy*

A host-based IDS is installed directly into a host whereas a network-based IDS is placed in the network to have a global view and detect attacks that target the network as a whole. Signature-based IDSs are based on specific attack patterns and are very good at detecting known attacks. They also have low false positive rates. However, they have a hard time detecting zero-day attacks. One of the most popular open-source signature-based IDS is snort. Anomaly-based IDSs base their detection on behaviors. They build-up a normal behavior for the network and report any unusual or suspicious activities. A hybrid IDS is one that combines both signature-based and anomaly-based techniques.

B. POX CONTROLLER

SDN offers a wide range of controllers. They range from commercial to open source ones. POX is a python based open source SDN controller. Our machine learning algorithm (Classification and Regression Trees in Python) had a huge impact on choosing this controller. POX comes by default with the Mininet Virtual Machine (VM) and needs no extra setup to make it work. Using POX, we can create our own modules for testing purposes. It works perfectly with OpenFlow which is the primary and most well-known southbound API [14].

C. DATA COLLECTION

Based on their sources, datasets can be classified into three groups: Public Datasets, Private Datasets, and Network Simulation Datasets [15]. Network simulation datasets are created by simulating normal and attack traffic by considering various attack scenarios. In this paper, we are using a network simulation dataset which we created by extracting three features (Duration, Packet Count, and Byte Count) from the flow statistics sent by the switches. The biggest advantage of our approach is that we can easily update our algorithm by simulating new attacks and retraining the algorithm with the new dataset.

D. DECISION TREES ALGORITHM

Decision Trees (DTs) are supervised learning algorithms used for both classification and regression problems. They are easy to understand and interpret, can be visualized, and can handle both numerical and categorical data.

Some DT algorithms are ID3, C.45, C5.0 and CART. Ross Quilan developed ID3 in 1986. To construct a tree, the algorithm divides attributes into two groups: the most dominant and the others. Entropy and information gains of each attribute is then calculated. The latest version release of Quinlan’s is C5.0 which is more accurate. Scikit-learn supports the Classification and Regression Trees (CART). Leo Breiman, a statistician at the University of California, Berkeley contributed greatly in the development of this algorithm. As its names indicates, this algorithm is capable of doing both classification and regression. The CART algorithm provides a basis for other algorithms such random forests.

One disadvantage of decision trees is that sometimes they could generate over-complex trees that do not generalize the data well.

III. Experimental Studies And Results

Using Mininet, we created a network with three switches and four hosts. Our machine learning algorithm is embedded into the POX controller which comes with Mininet pre-installed. The different components are then run to simulate the attack scenario and show how the system is capable of detecting and blocking attacks.

A. MININET

Mininet [16] is a network emulation tool for rapid prototyping of software defined networks. With a single command, Mininet allows us to create a virtual network running real kernel. This tool is useful for teaching and research. It is probably the best way to learn about SDN and the OpenFlow protocol. With it, we can easily interact with our network using an API or the Command Line Interface (CLI). Mininet is a fast tool that runs real programs and allows the customization of packet forwarding. It can be run on a laptop in a virtual machine.

Mininet has its limitations as well. These include resource consumption, usage of a single Linux kernel for all virtual hosts, and isolation from our LAN and internet (We may use NAT to connect Mininet to our LAN).

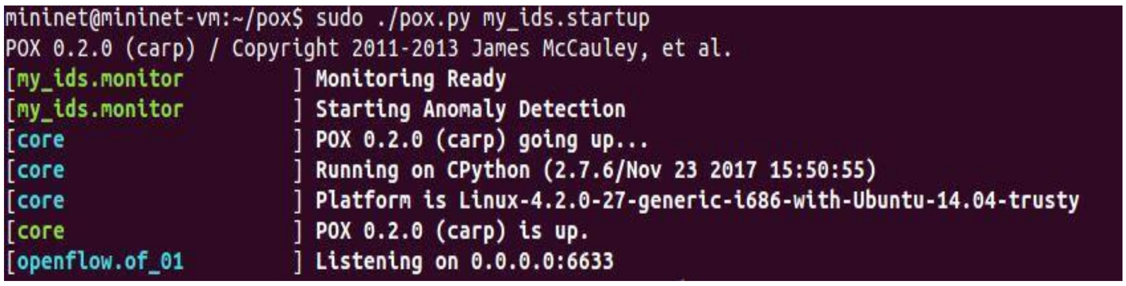
B. *OUR MACHINE LEARNING ALGORITHM*

We built the algorithm based on python Scikit-learn. The algorithm was trained using the dataset we created earlier. The classifier we chose for this project is decision trees. We fit the classifier into the training set. One of the biggest advantages of Scikit-learn in that classifiers can easily be changed with a single line of code. If we want to, let's say, train our algorithm with the Random Forest algorithm, we can just import its library and change one line of code.

Making predictions is as simple as doing classification. Using a module named joblib, the model can be persisted. The goal of model persistence is to save the model in a format that will allow us to import it to our particular project later on.

C. *ENVIRONMENT*

To lunch the system, we will need to first launch the POX controller and the IDS Module that will allow us to detect the attacks as seen in Figure 4. The IDS Module starts monitoring the network for attacks as soon as the controller is launched. The controller is listening on port 6633.



```
mininet@mininet-vm:~/pox$ sudo ./pox.py my_ids.startup
POX 0.2.0 (carp) / Copyright 2011-2013 James McCauley, et al.
[my_ids.monitor         ] Monitoring Ready
[my_ids.monitor         ] Starting Anomaly Detection
[core                   ] POX 0.2.0 (carp) going up...
[core                   ] Running on CPython (2.7.6/Nov 23 2017 15:50:55)
[core                   ] Platform is Linux-4.2.0-27-generic-i686-with-Ubuntu-14.04-trusty
[core                   ] POX 0.2.0 (carp) is up.
[openflow.of_01         ] Listening on 0.0.0.0:6633
```

Figure 4. *Launching the POX Controller*

We then create a tree topology network and connect it remotely to the controller on port 6633. In Mininet, a tree topology with a depth of two creates a network with four hosts, three switches, and one controller. Figure 5 depicts the topology created in Mininet.

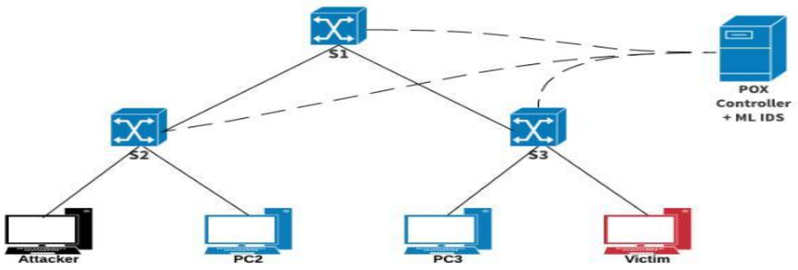


Figure 5. *Topology Created in Mininet*

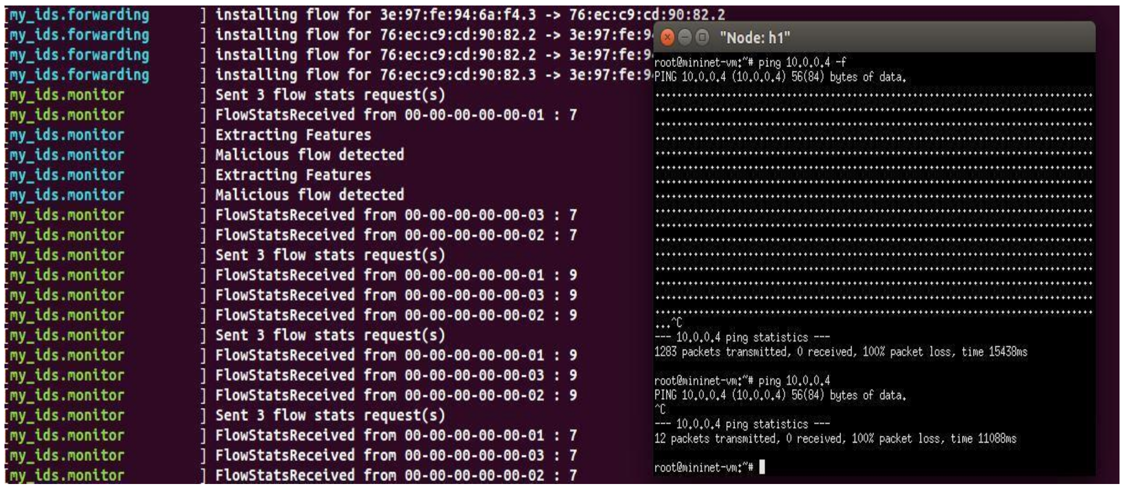
The switches communicate with the POX controller through the OpenFlow protocol. Flow statistics are kept into the flow tables of the switches. The ML IDS (Machine Learning based Intrusion Detection System) module of our system sends flow stats requests using POX's ofp_flow_stats_request messages and the three switches reply by sending it the flow statistics. Three features (Duration, Packet Count, Byte Count) are then extracted and forwarded to the flow-based IDS module in which the machine learning algorithm is loaded. As the algorithm has been trained using these three features, it is now able to predict new flows as either benign or malicious.

D. DDOS FLOOD ATTACK SCENARIO

Initially the attacker sends a normal ping. The switch checks its flow table, and because the destination is unknown to the switch (table miss), it forwards the packet to the POX controller (packet-in) to ask the path to send the packet. The SDN controller replies with a packet-out which includes the information needed by the switch. The switch then stores the destination of the route into its flow table. To carry out a ping flooding attack, the attacker will then send tens of thousands of packets per second to the server, flooding it with unnecessary echo request packets.

E. RESULTS

After setting up and running the environment, we execute a ping flood attack to test our algorithm. The attacker (10.0.0.1) sends intermittent pings as quickly as possible to the victim (10.0.0.4) using the -f argument in the command.



```
[my_ids.forwarding      ] installing flow for 3e:97:fe:94:6a:f4.3 -> 76:ec:c9:cd:90:82.2
[my_ids.forwarding      ] installing flow for 76:ec:c9:cd:90:82.2 -> 3e:97:fe:9
[my_ids.forwarding      ] installing flow for 76:ec:c9:cd:90:82.2 -> 3e:97:fe:9
[my_ids.forwarding      ] installing flow for 76:ec:c9:cd:90:82.3 -> 3e:97:fe:9
[my_ids.monitor         ] Sent 3 flow stats request(s)
[my_ids.monitor         ] FlowStatsReceived from 00-00-00-00-00-01 : 7
[my_ids.monitor         ] Extracting Features
[my_ids.monitor         ] Malicious flow detected
[my_ids.monitor         ] Extracting Features
[my_ids.monitor         ] Malicious flow detected
[my_ids.monitor         ] FlowStatsReceived from 00-00-00-00-00-03 : 7
[my_ids.monitor         ] FlowStatsReceived from 00-00-00-00-00-02 : 7
[my_ids.monitor         ] Sent 3 flow stats request(s)
[my_ids.monitor         ] FlowStatsReceived from 00-00-00-00-00-01 : 9
[my_ids.monitor         ] FlowStatsReceived from 00-00-00-00-00-03 : 9
[my_ids.monitor         ] FlowStatsReceived from 00-00-00-00-00-02 : 9
[my_ids.monitor         ] Sent 3 flow stats request(s)
[my_ids.monitor         ] FlowStatsReceived from 00-00-00-00-00-01 : 9
[my_ids.monitor         ] FlowStatsReceived from 00-00-00-00-00-03 : 9
[my_ids.monitor         ] FlowStatsReceived from 00-00-00-00-00-02 : 9
[my_ids.monitor         ] Sent 3 flow stats request(s)
[my_ids.monitor         ] FlowStatsReceived from 00-00-00-00-00-01 : 7
[my_ids.monitor         ] FlowStatsReceived from 00-00-00-00-00-03 : 7
[my_ids.monitor         ] FlowStatsReceived from 00-00-00-00-00-02 : 7
```

"Node: h1"

```
root@mininet-vm:~# ping 10.0.0.4 -f
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
...........................................................................
...........................................................................
...........................................................................
...........................................................................
...........................................................................
...........................................................................
...........................................................................
...........................................................................
...........................................................................
...........................................................................
...........................................................................
...........................................................................
...........................................................................
...........................................................................
...........................................................................
...........................................................................
...^C
--- 10.0.0.4 ping statistics ---
1283 packets transmitted, 0 received, 100% packet loss, time 15438ms

root@mininet-vm:~# ping 10.0.0.4
PING 10.0.0.4 (10.0.0.4) 56(84) bytes of data.
^C
--- 10.0.0.4 ping statistics ---
12 packets transmitted, 0 received, 100% packet loss, time 11088ms

root@mininet-vm:~#
```

Figure 6. *Ping Flood Attack Detection and Mitigation*

As seen in Figure 6, as soon as the flooding begins, the IDS shows in its log messages that a malicious flow has been detected and the POX controller is triggered to take an action and drop subsequent packets from the attacking host. Our proposed algorithm has successfully detected ping flooding in an SDN environment using machine learning. Thus, we have proved that machine learning has its place in SDN Security.

IV.CONCLUSION

In this paper, we proposed an IDS that allows us to detect ping flood attacks in an SDN network. The results show that choosing a network simulation dataset has its advantages as it allowed us to detect attacks. Machine learning with Scikit-learn gives us more flexibility as the algorithm can be changed in literally two or three lines of code. The dataset can also be easily updated to try other or new types of attacks. Overall, despite POX still supporting only python2.7, it can also help in developing IDSs. However, the developers of POX should consider updating it to python3 and include support for newer versions of OpenFlow (as of this writing, POX still supports OpenFlow 1.0 only). Mininet is a powerful tool and probably the best way to learn about SDN. Through it, we can emulate networks and connect them to any types of controllers in no time. Machine learning has its benefits, and for SDN to become more powerful and flexible, more and more machine learning algorithms should be tested and implemented in SDN.

As part of future work, new attack vectors could be added and tested as well. Implementing the algorithm in a hybrid environment, such as implementing it directly into snort would be very interesting. This way, we will be able to do signature-based and anomaly-based intrusion detection simultaneously.

V. REFERENCES

[1] Jonathan Crane. (2017, October 09). "Outage Prevention: Taking Humans Out Of The IT Equation," *Forbes*, [Online]. Available https://www.forbes.com/sites/ciocentral/2012/10/22/outage-prevention-taking-humans-out-of-the-it-equation/#3603b7504dd1.

[2] Kathleen Hickey. (2017, October 09). "What's behind most data center outages? [Online]. Available: https://gcn.com/articles/2016/02/09/data-center-outages.aspx.

[3] Press Release. (2017, October 09). "Global Survey: Complexity, Change and Human Factors Cause Network Outages - The Data Center Journal," 2016 [Online]. Available: http://www.datacenterjournal.com/global-survey-complexity-change-human-factors-cause-network-outages/.

[4] J. Networks Inc. "What's Behind Network Downtime? Proactive Steps to Reduce Human Error and Improve Availability of Networks" May, 2008.

[5] Rachel King. (2017, October 09). "Amazon Web Services Outage Caused by Human Error: A Typo | Fortune," 2017 [Online]. Available: http://fortune.com/2017/03/02/amazon-cloud-outage/.

[6] E. Description, “Cisco Certified Network Associate,” 2016.

[7] ONF. (2018, June 13) “Software-Defined Networking (SDN) Definition - Open Networking Foundation.” [Online]. Available: https://www.opennetworking.org/sdn-definition/.

[8] ONF. (2018, June 05) “Single Point of Failure. Not. - Open Networking Foundation.” [Online]. Available: https://www.opennetworking.org/news-and-events/blog/single-point-of-failure-not/.

[9] E. Banks. (2018, June 05) “SDN FAQ | Network World.” [Online]. Available: https://www.networkworld.com/article/2167706/lan-wan/lan-wan-sdn-faq.html.

[10] A. Mestres *et al.*, “Knowledge-Defined Networking,” *ACM SIGCOMM Computer Communication Review*, vol. 47, no. 3, 2017

[11] Y. L. Chen, (2018, June 05) “OpenDaylight Summit 2016: OpenDaylight Machine Learning &amp; Artifici...” [Online]. Available: https://opendaylightsummit2016.sched.com/event/80Nz.

[12] R. Braga, E. Mota, and A. Passito, “Lightweight DDoS Flooding Attack Detection Using NOX / OpenFlow Network-Based Mechanisms Using SDN Network-Based Mechanisms Using SDN,” pp. 408–415, 2018.

[13] L. Dongsoo, “Improving Detection Capability of Flow-based IDS in SDN,” M.S. thesis, Department of Computer Science, KAIST University, Daejeon, South Korea, 2015.

[14] SDxCentral, (2018, June 02) “What are SDN Southbound APIs? - Where they are used.” [Online]. Available: https://www.sdxcentral.com/sdn/definitions/southbound-interface-api/.

[15] M. H. Bhuyan, D. K. Bhattacharyya and J. K. Kalita, "Network Anomaly Detection: Methods, Systems and Tools," in *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 303-336, First Quarter 2014.

[16] Bob Lantz, (2018, June 01) “Mininet.” [Online]. Available: http://mininet.org/download/.