Deep Learning Approach for Network Intrusion Detection in Software Defined Networking

(Invited Paper)

Tuan A Tang*, Lotfi Mhamdi*, Des McLernon*, Syed Ali Raza Zaidi* and Mounir Ghogho*†
*School of Electronic and Electrical Engineering, The University of Leeds, Leeds, UK.
†International University of Rabat, Morocco.
Email: eltat@leeds.ac.uk, l.mhamdi@leeds.ac.uk, d.c.mclernon@leeds.ac.uk, eenarz@leeds.ac.uk and m.ghogho@leeds.ac.uk.

***Abstract*—Software Defined Networking (SDN) has recently emerged to become one of the promising solutions for the future Internet. With the logical centralization of controllers and a global network overview, SDN brings us a chance to strengthen our network security. However, SDN also brings us a dangerous increase in potential threats. In this paper, we apply a deep learning approach for flow-based anomaly detection in an SDN environment. We build a Deep Neural Network (DNN) model for an intrusion detection system and train the model with the NSL-KDD Dataset. In this work, we just use six basic features (that can be easily obtained in an SDN environment) taken from the forty-one features of NSL-KDD Dataset. Through experiments, we confirm that the deep learning approach shows strong potential to be used for flow-based anomaly detection in SDN environments.**

***Index Terms*—software defined networking; SDN; intrusion detection; deep learning; network security**

I. Introduction

Traditional network architecture has remained mostly unchanged over the past few decades and proved to be cumbersome. Software Defined Networking (SDN) is an emerging architecture that is dynamic, manageable, cost-effective, and adaptable, making it ideal for the high-bandwidth, dynamic nature of today's applications. This architecture decouples the network control and forwarding functions enabling the network control to become directly programmable and the underlying infrastructure to be abstracted for applications and network services [1]. SDN and OpenFlow [2] are increasingly attracting researchers from both academia and industry. The advantages of SDN in various scenarios (e.g., the enterprise, the datacenter, etc.) and across various backbone networks have already been proven (e.g., Google B4 [3] and Huawei carrier network [4]). Various SDN controller software have been proposed by open source organizations or commercial corporations, such as NOX [5], Ryu [6] and Floodlight [7]. There are various commercial vendors supporting OpenFlow in their hardware switches (e.g., HP, Pronto, Cisco, Dell, Intel, NEC and Juniper). OpenFlow protocol is the first standard communication interface defined between the control and forwarding layers of the SDN architecture. OpenFlow uses the concept of flow to identify the network traffic and records its information by counters. A flow is a group of IP packets, with some common properties, passing a monitoring point in a specified time interval.

Although the capabilities of SDN (e.g., software-based traffic analysis, logical centralized control, global overview of the network, and dynamic updating of forwarding rules) make it easy to detect and to react to network attacks, the separation of the control and data planes introduces new attack opportunities and so SDN itself may be a target of some attacks. According to Kreutz et al. [8], there are seven threat vectors in SDN. Three of them are specific to SDN and relate to the controller, the control-data interface, and the control-application interface. Network Intrusion Detection System (NIDS) protects a network from malicious software attacks. Traditionally, there are two types of NIDS according to strategies to detect network attacks. The first one, signature-based detection, compares new data with a knowledge base of known intrusions. Despite the fact that this approach cannot recognize new attacks, this remains the most popular approach in commercial intrusion detection systems. The second one, anomaly-based detection, compares new data with a model of normal user behavior and marks a significant deviation from this model as an anomaly using machine learning. As a result, this approach can detect zero-day attacks that have never been seen before. The anomaly-based detection approach is usually combined with flow-based traffic monitoring in NIDSs. Flow-based monitoring is based on information in packet headers, so flow-based NIDSs have to handle a considerably lower amount of data compared to payload-based NIDSs. Machine learning is successfully used in many areas of computer science like face detection and speech recognition, but not in intrusion detection. In [9], Robin Sommer and Vern Paxson mentioned many factors that affect the use of machine learning in network intrusion detection. Recently, deep learning has emerged and achieved real successes. So far, deep learning has been used extensively in computer science for voice, face and image recognition. Deep learning is capable of automatically finding correlation in the data, so it is a promising method for the next generation of intrusion detection. Deep learning can be used to efficiently detect zero-day attacks and so we can acquire a high detection rate.

Based on the flow-based nature of SDN, we propose a flow-based anomaly detection system using deep learning. In this paper, we apply a Deep Neural Network (DNN) and use it for the NIDS model in an SDN context. We train and evaluate the model by using the NSL-KDD Dataset [10]. Through experiments, we find an optimal hyper-parameter for DNN and confirm the detection rate and false alarm rate. The model gets the performance with accuracy of 75.75% which is quite reasonable from just using six basic network features.

The rest of the paper is organized as follows. Related works are introduced in section 2. In section 3, we give a brief introduction to deep learning, our deep learning model and the NSL-KDD Dataset. The performance of the model is analyzed in section 4. Finally, we draw conclusions and propose some future work in section 5.

II. Previous Work

Flow-based intrusion detection is extensively researched nowadays. In [11], the authors propose a flow-based anomaly detection system based on a Multi-Layer Perceptron and Gravitational Search Algorithm. The system can classify benign and malicious flows with a very high accuracy rate. In [12], the authors proposed an NIDS using a one-class support vector machine for their analysis and got a low false alarm rate. In contrast to other work, the system is trained with malicious rather than with a benign network dataset. Intrusion detection mechanisms in traditional networks have been widely studied and can be applied to SDN. To secure the OpenFlow network, many anomaly detection algorithms also have been implemented in the SDN environment. By using the programability of SDN, the authors of [13] show that a programmable home network router can provide the ideal platform and location in the network for detecting security problems in a SOHO (Small Office/Home Office) network. Four prominent traffic anomaly detection algorithms (threshold random walk with credit based rate limiting (TRW-CB algorithm), rate limiting, maximum entropy detector and NETAD) are implemented in an SDN context using OpenFlow compliant switches and a NOX controller. Experiments indicate that these algorithms are significantly more accurate in identifying malicious activities in the SOHO network than the ISP (Internet Service Provider) and the anomaly detector can work at line rates without introducing any new performance overhead for the home network traffic.

The SDN architecture is a target for many kinds of attacks and potential Distributed Denial of Service (DDoS) attack vulnerabilities exist across the SDN platform. DDoS attacks are an attempt to make a machine or network resource unavailable to its intended users. DDoS attacks are sent by two or more people or bots. For example, an attacker can take advantages of the characteristic of SDN to launch DDoS attacks against the control layer, the infrastructure layer plane and the application layer of SDN. DDoS attacks are easy to launch, but difficult to guard against. With the development of the Internet, DDoS is growing substantially. One major reason is the emergence and development of botnets which are networks formed by bots or machines compromised by malware. According to a Q1 2016 "State of the Internet Security" report by Akamai [14], the number of DDoS attacks had increased 125.36% in Q1 2016 compared to the number of attacks in Q1 2015. A lightweight method for DDoS attack detection based on traffic flow features is presented in [15] with extraction of 6-tuple features: 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). Self Organizing Maps (SOMs) are used as the classification method. In order to improve the scalability of native OpenFlow, a new method combining OpenFlow and sFlow has been proposed in [16] for an effective and scalable anomaly detection and mitigation mechanism in an SDN environment. Trung et al. [17] combine hard thresholds of detection and fuzzy inference system (FIS) to detect risk of DDoS attacks based on the real traffic characteristic under normal and attack states. Three features are chosen for detecting the attack: Distribution of Inter-arrival Time, Distribution of packet quantity per flow and Flow quantity to a server. Other researchers also use many feature selection algorithms to increase the detection accuracy.

In this paper, we use a Deep Neural Network (DNN) for anomaly detection. Six basic features are chosen for detecting attacks: *duration, protocol_type, src_bytes, dst_bytes, count* and *srv_count.* So the key difference between our work and other papers is that we use simplex preprocessing and features extraction in the SDN context.

III. Methodology

A. Deep Learning Approach

In classical machine learning, important features of an input are manually designed and the system automatically learns to map the features to an output. In deep learning, there are multiple levels of features. These features are automatically discovered and they are composed together in various levels to produce outputs. Each level represents abstract features that are discovered from the features presented in the previous level.

In our experiment, we constructed a simple deep neural network with an input layer, three hidden layers and an output layer as described in Figure 1. The input dimension is six and the output dimension is two. The hidden layers contain twelve, six and three neurons respectively. Our model initiation parameters are setup with 10 for the batch size and 100 for the epoch. The learning rate will be decided in the experiment.

B. Dataset

The KDD Cup is the leading data mining competition in the world. The NSL-KDD Dataset was proposed to solve some inherent problems of the KDD Cup 1999 Dataset [18]. Although it is quite old and not a perfect representative of existing real networks, it is still a good reference to compare the NIDS models. It has been used in the past to evaluate the performance of NIDS by many researchers, so there are significant performance measurement results for comparison. There are 125,973 network traffic samples in the KDDTrain+ Dataset and 22,554 network traffic samples in the KDDTest+ Dataset. Each traffic sample has forty one features that are categorized into three types of features: basic features, content-based features, traffic-based features. Attacks in the dataset are

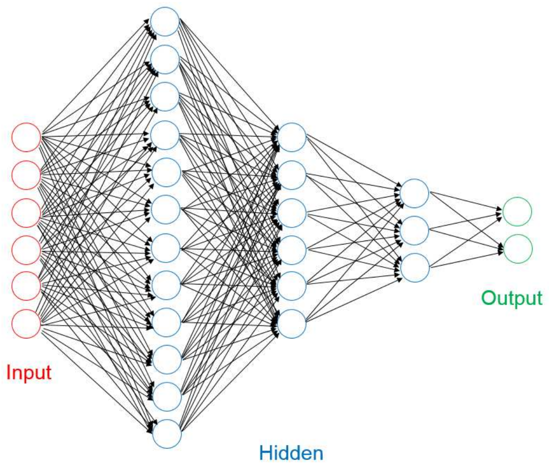

Fig. 1. Deep Learning Network Model

categorized into four categories according to their characteristics. The details of each category are described in Table I. Some specific attack types (written in bold) in the testing set do not appear in the training set. That makes the detection task more realistic.

TABLE I
ATTACKS IN THE NSL-KDD DATASET

Category	Training Set	Testing Set
DoS	back, land, neptune, pod, smurf, teardrop	back, land, neptune, pod, smurf, teardrop, **mailbomb, processtable, udpstorm, apache2, worm**
R2L	fpt-write, guess-passwd, imap, multihop, phf, spy, warezclient, warezmaster	fpt-write, guess-passwd, imap, multihop, phf, spy, warezmaster, **xlock, xsnoop, snmpguess, snmpgetattack, httptunnel, sendmail, named**
U2R	buffer-overflow, loadmodule, perl, rootkit	buffer-overflow, loadmodule, perl, rootkit, **sqlattack, xterm, ps**
Probe	ipsweep, nmap, portsweep, satan	ipsweep, nmap, portsweep, satan, **mscan, saint**

In our experiment, a subset of six features is chosen from the NSL-KDD Dataset having forty one features for training and testing. These six features are *duration, protocol_type, src_bytes, dst_bytes, count* and *srv_count*. Details of each feature can be found in Table II. These features are basic features and traffic-based features that can easily be obtained in SDN environments.

C. The SDN-based IDS Architecture

We propose an SDN-based NIDS architecture as depicted in Figure 2. In this architecture, the NIDS module is implemented in the controller. The SDN controller can monitor all the OpenFlow switches and request all network statistic when needed, so the NIDS module can take advantage of this global network overview for detecting intrusion. An *ofp_flow_stats_request* message will be sent from the controller to all the OpenFlow switches after a fixed time-window to request the network statistic. OpenFlow switches send back to the controller an *ofp_flow_stats_reply* message with all the statistics. The centralized controller can take advantages of the complete network overview supported by the SDN to analyze and correlate this feedback from the network. All network statistics will then be sent to the NIDS module for analysing and detecting any real time network intrusion. Once a network anomaly is detected and identified, the OF protocol can effectively mitigate it via flow table modification. New security policies can be propagated to switches in order to prevent attacks.

TABLE II
FEATURE DESCRIPTION

Feature Name	Description
duration	length (number of seconds) of the connection
protocol_type	type of the protocol, e.g. tcp, udp, etc.
src_bytes	number of data bytes from source to destination
dst_bytes	number of data bytes from destination to source
count	number of connections to the same host as the current connection in the past two seconds
srv_count	number of connections to the same service as the current connection in the past two seconds

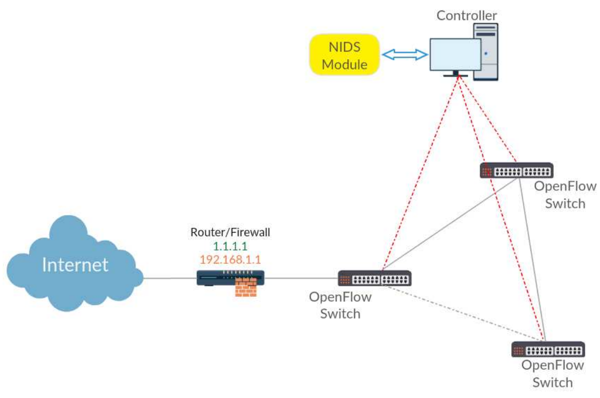

Fig. 2. Proposed SDN Security Architecture

IV. PERFORMANCE ANALYSIS

A. Evaluation Metrics

In general, the performance of NIDS is evaluated in terms of accuracy (AC), precision (P), recall (R) and F-measure (F). A NIDS requires high accuracy, high detection rate and low false alarm rate. A confusion matrix is used to calculate these parameters. In the confusion matrix, True Positive (TP) is the number of attack records correctly classified. True Negative (TN) is the number of normal records correctly classified. False Positive (FP) is the number of normal records incorrectly classified. False Negative (FN) is the number of attack records incorrectly classified. Then we can say:

- Accuracy (AC): shows the percentage of true detection over total traffic trace:

$$AC = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

- Precision (P): shows how many intrusions predicted by an NIDS are actual intrusions. The higher P is, the lower false alarm is:

$$P = \frac{TP}{TP + FP} \tag{2}$$

- Recall (R): shows the percentage of predicted intrusions versus all intrusion presented. We want a high R value:

$$R = \frac{TP}{TP + FN} \tag{3}$$

- F-measure (F): gives a better measure of the accuracy of an NIDS by considering both the precision (P) and the recal (R). We also aim for a high F value:

$$F = \frac{2}{\frac{1}{P} + \frac{1}{R}} \tag{4}$$

B. Experimental Results

Initially, we implemented the NIDS for 2-class classification (normal and anomaly class). The performance of the model depends on the value of the hype-parameter initiated. Firstly, the model was optimized by finding the hyper-parameter which leads to optimal classification results. Thus, we tried to optimize the model by varying the value of learning rate in the range {0.1, 0.01, 0.001, 0.0001}. When training a model, we tried to minimize the loss and maximize the accuracy. By comparing loss and accuracy of the training phase (see Table III), we can see that along with the decrease of the learning rate, the loss will decrease and the accuracy will increase. However, in the testing phase, when we decreased the learning rate to 0.0001, the results are not as good as the learning rate of 0.001. This is because if the learning rate is too small, the NIDS model will be trained too accurately. That is the reason for the best results in the training phase of the learning rate of 0.0001. Because of this too accurate training phase, the model cannot generalize the characteristic of the training samples well. So while the model can easily catch the intrusion instances in the training set, it cannot catch the new intrusion instances in the test set. As a result, the NIDS performance decreases in the testing phase.

TABLE III
Loss and Accuracy Evaluation for Different Learning Rates

Learning Rate	Train Set		Test Set	
	Loss (%)	Accuracy (%)	Loss (%)	Accuracy (%)
0.1	11.49	88.04	31.26	72.05
0.01	8.41	90.9	20.15	73.03
0.001	8.26	91.62	19.51	75.75
0.0001	7.45	91.7	20.3	74.67

Secondly, we evaluated the precision, recall, and f-measure of the model for more details. The performance of the DNN algorithm was evaluated using the test data provided. The performance of the model with each learning rate is shown in Table IV. As we can see, the learning rate of 0.001 gives us the best results among four learning rates in all evaluation metrics. All evaluation metrics are in a growing trend when we decrease the learning rate from 0.1 to 0.001, but the metrics suddenly decrease at a learning rate of 0.0001. From evaluating the accuracy metrics, loss and accuracy, we can see that the performance of the NIDS model decreases when the learning rate is decreased to 0.0001.

TABLE IV
Accuracy Metrics for Different Learning Rates

Learning Rate	Precision (%)	Recall (%)	F1-score (%)
0.1	79	72	72
0.01	82	73	72
0.001	83	76	75
0.0001	83	75	74

In the following, Receiver Operating Characteristic (ROC) curves are presented, with respect to true positive rate and false positive rate, for each of the learning rates. The area under the ROC Curve (AUC) is a standard measure for classifier comparison. The AUC is calculated via the "trapezoidal" rule, where a trapezoid is constructed from the lines drawn for each two consecutive points on the curve. The higher the ROC curve's area, the better the system. Figure 3 shows the ROC curves for four different learning rates. As expected, the learning rate of 0.001 performs better than others with the highest AUC. As we can see, the learning rate of 0.0001 has a slightly poorer performance than that of the learning rate of 0.001. However, the learning rate of 0.0001 has a lower false positive rate than the learning rate of 0.001. This is because the model was trained quite accurately with a low learning rate. For overall evaluation, the learning rate of 0.001 has the best performance, so it is chosen for further evaluations.

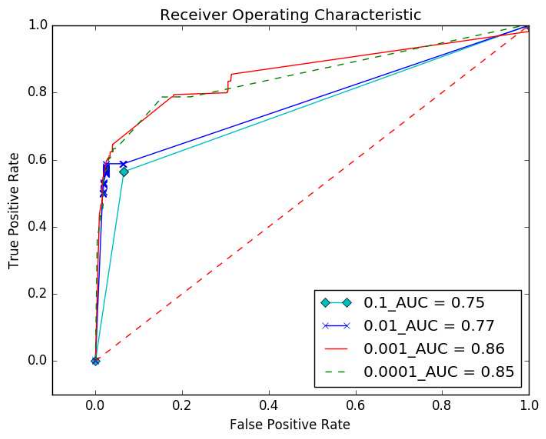

Fig. 3. ROC Curve Comparison for Different Learning Rates

We also evaluated our work by comparing the results with that of other works and other machine learning algorithms. In [15], the authors use 6-tuple features: APf, ABf, ADf, PPf, GSf and GDP for DDoS detection. Their result is good with a detection rate of 99.11% and a false alarm rate of 0.46%. Our results are quite low compared to their results. The main reason may be our feature selection for training and testing. Their six features are mainly specified for detecting DDoS attacks. In our case, we just chose the six most basic features and we did not focus on any kind of attacks. In our future research, we will try to apply their 6-tuple features to our model for further evaluation. We compared our results with the result in [10] from different machine learning algorithms. In [10], the authors train and test different algorithms with a full training and testing set of forty one features. From these experiments, the authors can evaluate the performance of these algorithms in their dataset. From Table V, we can see that our DNN approach is quite low compared with other algorithms with 75.75% accuracy. The most accurate machine learning algorithm is the random tree with 81.59%. This is the accuracy obtained from full feature training set, and so the random tree algorithm can generalize the characteristics of the normal and anomaly traffic very well. In our simulation, it must be noted that only the six basic features which are mentioned in Table I are used for training and testing. This sub-feature set cannot provide enough information for our DNN algorithm to generalize the characteristics of some sophisticated or new attacks. However, it can be seen that algorithm performs reasonably compared with other algorithms.

TABLE V
ACCURACY COMPARISON OF DIFFERENT ALGORITHMS

Algorithm	**Accuracy (%)**
J48	81.05
Naive Bayes (NB)	76.56
NB Tree	82.02
Random Forest	80.67
Random Tree	81.59
Multi-layer Perceptron	77.41
Support Vector Machine (SVM)	69.52
Our DNN	**75.75**

For further evaluation, we just use the sub-feature set for training and testing. In the next experiment, the proposed DNN algorithm in this study was compared with other machine learning algorithms: NB, SVM and Decision Tree. Table VI gives us an overview about the performance of each machine learning algorithm using sub-feature dataset. The proposed DNN algorithm gives us the best results amongst all algorithms. The other machine learning algorithms cannot generalize the characteristic of training samples well with just six features, so they achieve quite poor performance.

TABLE VI
ACCURACY COMPARISON FOR SUB-FEATURE DATASET

Algorithm	**Accuracy (%)**
Naive Bayes	45
SVM	70.9
Decision Tree	74
Our DNN	**75.75**

Finally, the ROC curves of these four algorithms are presented in Figure 4. As expected, deep learning algorithm performances are better than others with the highest AUC. Our DNN algorithm achieves a higher accuracy rate with a lower false positive rate compared to others.

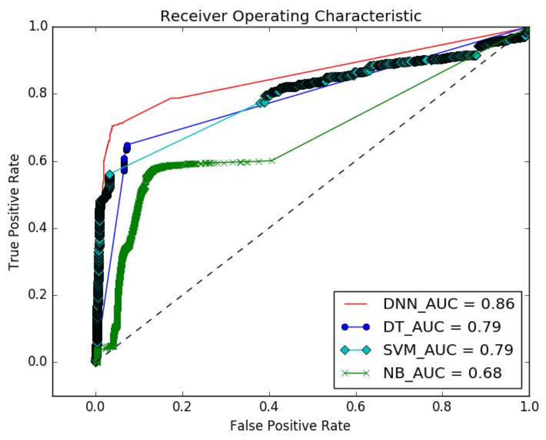

Fig. 4. ROC Curve Comparison for Different Algorithms

From all the above, we demonstrated that our proposed DNN approach can generalize and abstract the characteristics of the normal and anomaly traffic with a small number of features and gives us a promising accuracy.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we have implemented a deep learning algorithm for detecting network intrusion and evaluated our NIDS model. Although our results are not yet good enough to be adopted in any commercial product or an alternative solution for signature-based IDS, our approach still has significant potential and advantages for further development. By comparing the results with those of other classifiers, we have shown the potential of using deep learning for the flow-based anomaly detection system. In the context of the SDN environment, the deep learning approach also has potential. This is attributed to the centralized nature of the controller and the flexible structure of SDN. The basic information about network traffic can be extracted easily by the controller an evaluated by the deep learning intrusion detection module. To improve the accuracy, we will analyze the traffic and propose other types of features. With the flexibility of the SDN structure, we can extract many features that contain more valuable information or focus on one specific type of attack, like DDoS, to increase the accuracy of the NIDS. We will also try to adjust our DNN model for better performance (e.g., varying the number of hidden layers and hidden neurons). In the near future, we will also try to implement this approach in a real SDN environment with real network traffic and evaluate the performance of the whole network in terms of latency and throughput.

REFERENCES

[1] "Software Defined Networking Definition," Available: https://www.opennetworking.org/sdn-resources/sdn-definition, [Accessed 04 Jul. 2016].

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[3] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a globally-deployed software defined wan," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 3–14, 2013.

[4] C. T. Huawei Press Centre and H. unveil world's first commercial deployment of SDN in carrier networks, "[online]. available: pr.huawei.com/en/news/hw-332209-sdn.htm."

[5] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "Nox: towards an operating system for networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 3, pp. 105–110, 2008.

[6] "Ryu," Available: http://http://osrg.github.io/ryu/.

[7] "Floodlight," Available: http://www.projectfloodlight.org/.

[8] D. Kreutz, F. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*. ACM, 2013, pp. 55–60.

[9] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *2010 IEEE symposium on security and privacy*. IEEE, 2010, pp. 305–316.

[10] M. Tavallaee, E. Bagheri, W. Lu, and A.-A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications*, 2009.

[11] Z. Jadidi, V. Muthukkumarasamy, E. Sithirasenan, and M. Sheikhan, "Flow-based anomaly detection using neural network optimized with gsa algorithm," in *2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops*, 2013, pp. 76–81.

[12] P. Winter, E. Hermann, and M. Zeilinger, "Inductive intrusion detection in flow-based network data using one-class support vector machines," in *New Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference on*. IEEE, 2011, pp. 1–5.

[13] S. A. Mehdi, J. Khalid, and S. A. Khayam, "Revisiting traffic anomaly detection using software defined networking," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2011, pp. 161–180.

[14] "Q1 2016 State of the Internet / Security Report," Available: https://content.akamai.com/PG6301-SOTI-Security.html, [Accessed 07 Jul. 2016].

[15] R. Braga, E. Mota, and A. Passito, "Lightweight ddos flooding attack detection using nox/openflow," in *Local Computer Networks (LCN), 2010 IEEE 35th Conference on*. IEEE, 2010, pp. 408–415.

[16] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on sdn environments," *Computer Networks*, vol. 62, pp. 122–136, 2014.

[17] P. Van Trung, T. T. Huong, D. Van Tuyen, D. M. Duc, N. H. Thanh, and A. Marshall, "A multi-criteria-based ddos-attack prevention solution using software defined networking," in *Advanced Technologies for Communications (ATC), 2015 International Conference on*. IEEE, 2015, pp. 308–313.

[18] "KDD Cup 1999," Available: http://kdd.ics.uci.edu/databases/kddcup99/, [Accessed 04 Jul. 2016].