Sule Lamido University Journal of Science and Technology (SLUJST) Vol. 2 No. 1 [January, 2021], pp. 45-52

An Adam-Optimised Deep Neural Net for Predicting Vulnerability Attack in Software Defined Networking

Umar Umar Abdullahi[1], Isa Modibbo Ismail[2]*, Babangida Isyaku[3]
[1,2,3]Mathematics and Computer Science, Sule Lamido University Kafin Hausa, Nigeria
umarabdullahimarson@gmail.com, imodibbo@slu.edu.ng, babangida.isyaku@slu.edu.ng

Abstract

Software Defined Networking (SDN) has simplified network management through the separation of the control plane from the data plane. SDN controller decides where traffic flow is forwarded while the data plane, composed of network switches that forwards flow packets based on the decision made by the control plane. Forwarding logic is stored on a specialized switch memory of limited capacity known as Ternary Content Addressable Memory (TCAM). The centrality of control and the limitation of the flow table are the two most vulnerable aspects of SDN. The former is exploited by flooding the controller with malicious requests to overburden the controller and pave way for malicious attacks while the latter involves exploiting the limited flow table overflow. This paper proposes an Adam-optimised Deep Neural Network-based model for predicting these two vulnerability attacks in Software Defined Networks setting. This approach was tested on the NSL-KDD dataset, achieving an accuracy score of 93%. Experimental results also showed that this approach exhibited favorable performance on other metrics relative to some popular Machine Learning techniques. We conclude that this approach shows strong potential for Adam-optimised Deep Learners for SDN vulnerability attack mitigation.

Keywords: Software Defined Networking, Intrusion Detection, Neural Network, Feature Selection

1. Introduction

Advancement of technology has rapidly increased access to the Internet with increasingly powerful devices. As such, the number of internet-connected devices and traffic flow volume has significantly increased. According to a recent study, the annual global network traffic flow as of 2018 reached 19.01 exabytes a month and is predicted to reach 77.5 exabytes by 2022 globally (J. Clement, 2020). As a result, meeting up the current network demand using a conventional (IP) based network is extremely difficult. A leading contributory reason for that is the vertical integration of the traditional network control plane with the data plane. Software Defined Networking (SDN) based design has been shown to be more adaptable, scalable, robust, and equally important, energy-efficient, than traditional networking (Isyaku et al., 2020).

SDN emerged to decouple the control plane of a network from its data plane. The Control plane composes of one or more controllers. Data plane consists of a set of OpenFlow switches that are used to forward packets to the desired destination. SDN controller is the brain of the network that decides where and how to forward traffic. In addition, it can maintain the global network state and can manipulate the entire network through the flow entries installed in the OpenFlow switch flow table (Zahid et al., 2017). Incoming traffic flows are processed according to the control logic installed in the switch flow table.

Flow table entries in switches are installed in two modes: proactive and reactive. In the latter, entries are installed in the flow table based on demand and upon arrival. As a result, an increase in the number of traffic flows will surge the processing load on the controller because of the need to process every flow and install the corresponding entry (Isyaku et al., 2020). This easily exhausts the limited bandwidth allocation between the controller and switch thereby reducing overall Quality of Service (QoS). As for the latter, flow entries are installed in advance before the arrival of flow. However, the number of flows is usually huge in an enterprise network. According to (Yang et al., 2016) there can be up to 200k flows arrival/sec for a data center with 4k server. This way the

corresponding flows need to be stored in the switch flow table. Unfortunately, the switch flow table memory is available in limited space and therefore, cannot accommodate such a dense number of flows. Typical OpenFlow switch flow tables store rules in a specialized high-speed memory called ternary content addressable memory (TCAM) which provides O(1) time flow entry lookup each clock cycle. Despite the high-speed lookup, TCAM memory only holds a few thousand entries (Ahmed et al., 2017).

The central controller and switch flow table memory limitation are the new security vulnerabilities of the SDN (Correa Chica et al., 2020). Some of the most common threats to the SDN are Distributed Denial of Service (DDoS) and Man in the Middle (MIM) attacks. This is usually launched against the controller or the switch memory (Polat et al., 2020). An example scenario involves an attacker generating a large amount of fake traffic to the switch, such that if a legitimate flow arrives, and no corresponding entry is found, a table miss occurs. This leverages the limited size of the flow table, generating a Denial of Service (DoS) attack thereby preventing writing rules for legitimate clients. In any case, predicting and classification of attacks is imperative to properly mitigate security threats in SDN. However, this is more challenging than in traditional networks (Dridi & Zhani, 2018). This called for the need to employ machine learning techniques to classify different security threats. This paper presents an optimized deep learner classification system for improving security at the SDN controller, which is designed to effectively detect attacks due to the major vulnerabilities outlined above.

The rest of the paper is organized as follows: Section 2 shows related work. In Section 3, we present the proposed model. Section 4 discusses the evaluation of the proposed model including a comparison of its performance with other ML-based SDN detectors. Finally, section 5 concludes the paper and provides future work.

2. Related Work

The adoption of SDN technology to industries and carrier-grade networks is rapidly increasing. Investigating security threats, vulnerabilities, and how to mitigate SDN security concerns have widely gain researcher's attention from both industry and academia lately (Elsayed et al., 2019; Zhang et al., 2018). These concerns are investigated in two ways; the first category focuses on addressing the flow table overflow problem due to its memory limitation and secondly central SDN controller. These are mostly the new target for attack in SDN. Other efforts focused on how to mitigate DDoS attacks on the central controller and the switch flowtable. Yet other solutions tried to classify different possible attacks on SDN and how to overcome it (Elsayed et al., 2019).

In an attempt to reduce the flow table overflow, some works proposed utilization of a timeout (Lu et al., 2016) to control the life span of flow entry to match the event generated to the controller at a discrete-time interval. (Zhou et al., 2018) focused on improving network performance by reducing the flow table overflow rate used by attackers to launch an attack and subsequently bring the network down.

The work of (Luo et al., 2018) analyzed the impact of timeout on flow-table performance and proposed dynamic eviction algorithm for flow entries thereby mitigating the effect on flow table and improving the network performance. (Isyaku et al., 2020) proposed an idle and hard timeout allocation algorithm (IHTA) that assigns different timeout based on traffic flow behavior. This is to efficiently improve the usage of flow table memory by removing the flow entries that never repeat themselves more often. In this way memory space is preserved and chances of overflow are reduced. (Kim et al., 2017) proposed that expired traffic flows were given a second chance to avoid consulting controller.

Other works focus on security monitoring such as access control list (ACL), adaptive Firewalls, DoS and DDoS. For example: (Yan et al., 2018) studied SDN in a cloud computing environment, specifically, on the Distributed Denial of Service (DDoS) attacks. However, the programmability of SDN offers many options to address network attacks when they are detected. As part of the effort to mitigate the effect of intrusions in SDN, (Song et al., 2017) devised a threat-aware system proposal that detects and responds to network intrusion. The system pre-processes data first, then builds a predictive data model, followed by a decision-making stage.

Lately, the proposal of (Polat et al., 2020) trained historical network attack data using a machine-learning algorithm to identify possible malicious and prospective new targets to mitigate security threat vectors. Bayesian, Naive Bayes, and Decision Table algorithms were applied. There have been Machine Learning experiments which have proved to be quite effective in detecting malicious users and threat classification in the SDN data plane. An example is the work of (Elsayed et al., 2020) that leverages on a machine learning model to tackle DDoS attack using the features selection method. Features were obtained from the SDN dataset under normal network conditions and attack situations. (Lee et al., 2017) used Auto-Encoders (AE) with Recurrent Neural Network (RNN) to build a threat classification model. The AE detects anomalies through multiple hidden encode/decode layers. The RNN, which is notable for its success in modeling sequences, served as the classification module. It also used entropy estimates to extract features and Support Vector Machine (SVM) to build a detection system for DDoS attacks in SDN. (Polat et al., 2020) explores SDN layers structure and proposed bidirectional recurrent neural network (RNN) model to detect and block DDoS attack. Although the model targeted real-time attack detection, it also had a high accuracy rate in blocking DDoS attacks. However, this may not be effective in a large scale network environment where multiple controllers may be required (Elsayed et al., 2019).

Prior studies have used various datasets to validate their detection method. However, obtaining all features for effective attack classification from SDN central controller proved expensive. More recently, NSL-KDD has been effectively used in the analyses of SDN security concerns and is widely used for academic research. (Elsayed et al., 2019) used PCA for feature reduction on NSL-KDD because t-SNE is noisy and slow. The model was applied with SVM, J48, Naive Bayes, Random Forest algorithms, and were shown to have low precision, recall, F-score & accuracy scores.. Deep learner classifiers detectors include (Tang et al., 2018) that used a Gated Recurrent Unit Recurrent Neural Network (GRU-RNN) (a form of deep RNN) to build an anomaly detection system.

While research researches are better with time, there is still growing demand for more effective machine learning security threat, vector, classification in SDN with high accuracy to detect the security challenge in SDN controller and Flow table security concern. Such a proposal should be easy to implement in a large scale network environment. In contrast to the existing studies, this paper proposed an SDN targeted Deep Neural Network that models complex network parameter features relationships, used on optimally experimental Neural Network model parameters, The model system is to deployed for vulnerability detection in high traffic and volume SDN controller.

3. Proposed Method

This section presents the architecture of the proposed method. It consists of an SDN-tuned Deep Neural Network (DNN) model designed with an SDN environment in mind.

Figure 1 shows the architecture of our proposed model. Complex features relationship learning starts with features selection using the highly optimized Principal Component Analysis (PCA) algorithm. The DNN model parameters (number of neurons and depth of hidden layer) where experimentally chosen to facilitate faster convergence in order to handle the high-velocity setting. The DNN model takes input from an optimized feature obtained from the PCA selection.

The log-loss function of the model was optimized using the Adaptive Moment Estimation Algorithm (Adam) optimisation algorithm (Kingma & Ba, 2014), which is essential for a limited memory setting as is typical in SDN flow table and improves performance with large datasets. The Adam algorithm is presented in Figure 2. This optimises the weights of the nodes during training. The multi-label classifier model DNN model is passed through a simple binary-classifier function (attack vs normal flow) that identifies vulnerability classes from others.

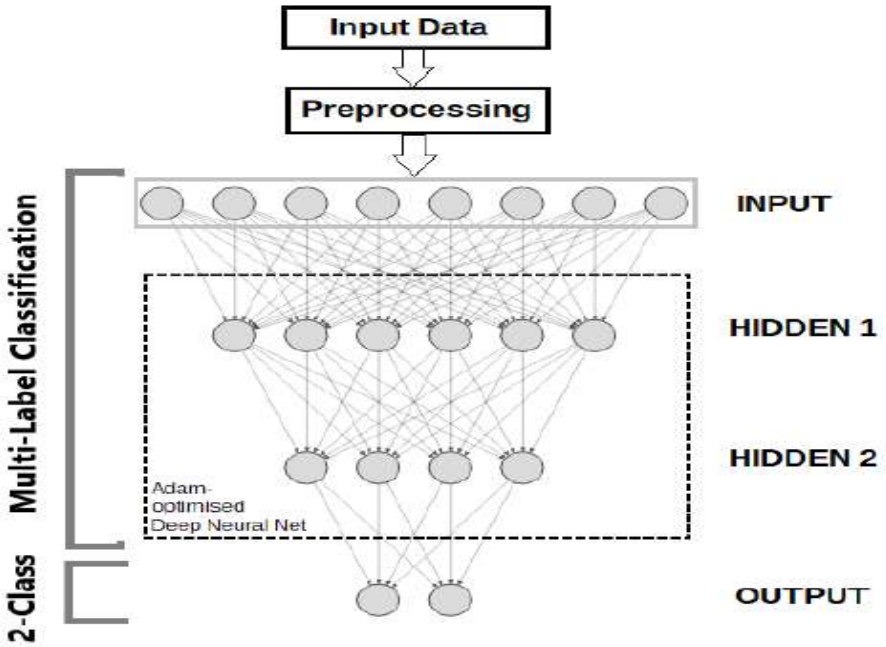

FIGURE 1. The Proposed Method Architecture

An Andrew curve plot of the benchmark dataset chosen (NSL-KDD) by (Elsayed et al., 2019) on the NSL-KDD shows a high-feature space non-linearity of high-degree. This further supports a Deep learning based solution for the Open-Flow vulnerability problem.

ADAM$(f, \alpha, \beta_{1,2}, \theta, \epsilon, g_t)$

1. //First initialisations
2. $m_0 \leftarrow 0$ (1^{st} moment vector)
3. $v_0 \leftarrow 0$ (2^{nd} moment vector)
4. $t \leftarrow 0$ (timestep initialisation)
5. **while** θ_t is not converged **do**:
6. $\quad t \leftarrow t + 1$
7. $\quad g_t \leftarrow \Delta_\theta f_t(\theta_{t-1})$
8. $\quad m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$
9. $\quad v_t \leftarrow \beta_2 \cdot m_{t-1} + (1 - \beta_2) \cdot g_t^2$
10. $\quad \hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$
11. $\quad \hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$
12. $\quad \theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$
13. **end while**
14. **return** θ_t (resulting parameter)

FIGURE 2. The ADAM stochastic optimisation algorithm

This work proposes an SDN targeted Deep Neural Network that models complex network parameter features relationships, used with optimal hyper-parameters Neural Network model, providing predictiveness and adaptivity required in a real-life SDN environment setting. The model system is deployed within the SDN controller for vulnerability detection in a high traffic and high volume setting.

4. Evaluation

4.1 Dataset

According to (Nguyen, 2018), a major challenge of Machine Learning based SDN solutions is the near scarcity of organic training data, instead, there is a high reliance on synthetic data. The shortage of publicly available datasets has been argued to be due to privacy and legal concerns that limit publications of such network data.

The KDD-99 dataset has been the benchmark dataset for intrusion systems evaluation. However, works such as (Tavallaee et al., 2009) showed its inherent problems and proposed the NSL-KDD. The NSL-KDD did showed great improvement over KDD-99, reducing a lot of redundancies. However, it also inherited some of its problems. Nevertheless, it has been taken as the de factor research benchmark dataset by all researchers (Elsayed et al., 2019),, maintaining a dominant status in general network intrusion research, making it effective for comparison of research methods. This work uses NSL-KDD to evaluate the proposed method.

The NSL-KDD contains network traffic data records of 41 features and 21 attack patterns. The distribution of the attack categories and types is shown in Table 1. Only 10% of the database is used for final testing. The other 90% was put for 10-fold cross-validation training.

4.2 Experimental Setup

The 41 features NSL-KDD dataset is first cleaned by removing missing values (denoted as NaN) and infinity symbol (-). The Principal Component Analysis algorithm was used to select the most representative features. Hyper-parameter values were experimentally determined. Experiments were conducted to determine optimal values. Firstly, the performance of the model was tested using different, albeit increasing, learning rates values. The performance metric and results are shown in Figure 2, this showed that better performance is gotten with a lower learning rate. However, the problem with low learning rates is that they converge slowly, which is generally a concern for the SDN application setting.

Additionally, we also experimented to determine the optimal hidden layers, iteration number, and activation functions. Optimal performance was found using seven layers as performance did not increase with more hidden layers even though training time increased significantly.

The training dataset was then split into 10-folds of equal sizes for 10-cross validation using learned hyper-parameters: at ten iterations, each fold serves to validate training on the other four-folds. The performance is then

TABLE 1. NSL-KDD attack categories distribution

Category	Attack		Set Count	
	Training	**Testing Additions**	**Training**	**Testing**
DOS	Back, land, Neptune, pod, smurf, teardrop	Mailbomb, processtable, udpstorm, apache2, worm	45,927	5,741
Probe	Ipsweep, nmap, portseep, satan	Mscan, saint	11,656	1,106
R2L	Fpt-write, guesspassword, imap, multihop, phf, spy, warezelient, warezmaster	Xlock, isnoop, snmpguess, snmpgetattack, httptunnel, sendmail, named	995	2,199
U2R	Buffer-overflow, loadmodule, perl, rootkit	Sqlattack, xtern, ps	52	37

averaged for the whole process.

4.3 Performance Evaluation

The confusion matrix, also an error matrix, is used to visualize the performance of a classification system. It presents a summary of false vs true predictions. The accuracy of our model is shown in Table 3, showing the

TABLE 2. Hyper-Parameter tuning: Learning rate performance

Learning Rate	Precision		Recall		F-Score		Accuracy (%)
	Attack	Benign	Attack	Benign	Attack	Benign	
0.1	0.89	0.90	0.90	0.90	0.90	0.90	90
0.01	0.88	0.90	0.89	0.89	0.89	0.89	89
0.001	0.89	0.89	0.89	0.90	0.89	0.89	89
0.0001	0.90	0.91	0.91	0.91	0.91	0.91	91
0.00001	0.90	0.91	0.90	0.91	0.91	0.91	91
0.000001	0.89	0.90	0.92	0.92	0.92	0.92	92
0.0000001	0.91	0.92	0.92	0.92	0.91	0.92	92
0.00000001	0.93	0.93	0.93	0.93	0.93	0.93	93

detection rate of both normal flow traffic and attack to be 0.93%, i.e. its classification performance is 93%.
Additionally, we used other standard metrics used in intrusion detection research to evaluate the performance of our model relative to other works. These include the precision, recall, F-score and accuracy measures, and defined as follows:

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

$$F - Score = \frac{2 \times Precision \times Recall}{Precision \times Recall} \quad (3)$$

$$Accuracy \quad \frac{TP + TN}{TP + TN + FP + FN} \quad (4)$$

where: TP = True Positive, FP = False Positive, TN = True Negative and FN = False Negative
Finally, we compared our model performance with that of a number of well-known Machine Learning techniques, outlined in (Elsayed et al., 2020). These include Booster, Decision Tree (DT), Logistic Regression (LR), NB, Random Forest (RF) and Support Vector Machine (SVM). Table 4 presents the comparison and shows that our model only comes behind LR in performance and has out-performed four techniques in overall detection accuracy. Our Atom-optimised DNN model will be preferred by far over the best performer, LR, considering F-scores.

TABLE 3. Confusion Matrix

Attack	**0.93**	**0.07**
Normal	0.07	0.93
	Attack	**Normal**

4.4 Discussion

The paper represents the potential of the Adam-optimisation application in Machine Learning based SDN security. It presented a novel Adam-based Deep Neural Net for (classifying/detecting) (attack/vulnerability-exploit) traffic input of the SDN controller. Our technique compared favorably to a lot of Machine Learning techniques. This partly due to the ability of deep learners to model complex relationships well, this makes them more suitable for general network intrusion detection when compared with traditional detection systems. Moreover, they are known to do better with big data, an emerging network threat platform. This model solution can be installed at the SDN controller application layer.

TABLE 4. Performance comparison with ML techniques

Techniques	Precision		Recall		F-Score		Accuracy (%)
	Attack	Benign	Attack	Benign	Attack	Benign	
LR	0.79	0.85	0.81	0.84	0.84	0.85	85
SVM	0.80	0.80	0.80	0.81	0.80	0.81	81
KNN	0.83	0.83	0.83	0.87	0.83	0.84	82
GNB	0.51	0.52	0.53	0.53	0.53	0.54	53
DT	0.79	0.80	0.79	0.79	0.78	0.80	78
RF	0.87	0.87	0.87	0.87	0.87	0.87	87
Proposed NN	**0.93**	**0.93**	**0.93**	**0.93**	**0.93**	**0.93**	**0.93**

5. Conclusion & Future work

Software Defined Networking offer increased robustness, scalability, and programmability but is also prone to new attack methods unknown to the traditional system. A major contributor to this is the limited capacity of the OpenFlow switch table, exposing the SDN network vulnerable to several attack types. This paper proposed an Adam optimized Deep Neural Network that integrates the limited capacity and high-velocity traffic of the SDN controller setting into vulnerability attack detection, thereby allowing the controller to deny the malicious request or forward for further investigation. The NSL-KDD dataset was used to train and validate the model. Evaluation results showed the proposed model to be promising, especially for high traffic SDN environments.

It is imperative to note that there is still a need to improve the performance of the proposed model for real-life deployment as Machine learning systems generally under-perform in the real-world setting (Sultana et al., 2019). Hence, we will like to see how we can improve the model performance in the future. We also intend to redesign the model to a multi-class detection system that can evaluate its performance on specific attack types. We are also interested in evaluating this model on other datasets especially those that can represent SDN-traffic more effectively.

Reference

Ahmed, A., Park, K., & Baeg, S. (2017). Resource-Efficient SRAM-Based Ternary Content Addressable Memory. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, *25*(4), 1583–1587. https://doi.org/10.1109/TVLSI.2016.2636294

Correa Chica, J. C., Imbachi, J. C., & Botero Vega, J. F. (2020). Security in SDN: A comprehensive survey. *Journal of Network and Computer Applications*, *159*(November 2019), 102595. https://doi.org/10.1016/j.jnca.2020.102595

Dridi, L., & Zhani, M. F. (2018). A holistic approach to mitigating DoS attacks in SDN networks. *International Journal of Network Management*, *28*(1), 1–14. https://doi.org/10.1002/nem.1996

Elsayed, M. S., Le-Khac, N.-A., Dev, S., & Jurcut, A. D. (2020). DDoSNet: A Deep-Learning Model for Detecting Network Attacks. *CoRR*, *abs/2006.1*. https://arxiv.org/abs/2006.13981

Elsayed, M. S., Le-Khac, N.-A., Dev, S., & Jurcut, A. D. (2019). Machine-Learning Techniques for Detecting Attacks in SDN. *2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*, 277–281. https://doi.org/10.1109/ICCSNT47585.2019.8962519

Isyaku, B., Kamat, M. B., Abu Bakar, K. Bin, Mohd Zahid, M. S., & Ghaleb, F. A. (2020). IHTA: Dynamic Idle-Hard Timeout Allocation Algorithm based OpenFlow Switch. *ISCAIE 2020 - IEEE 10th Symposium on Computer Applications and Industrial Electronics*, 170–175. https://doi.org/10.1109/ISCAIE47305.2020.9108803

J. Clement. (2020). *Global mobile data traffic 2017-2022*. https://www.statista.com/statistics/271405/global-mobile-data-traffic-forecast/

Kim, E. Do, Choi, Y., Lee, S. I., & Kim, H. J. (2017). Enhanced flow table management scheme with an LRU-based caching algorithm for SDN. *IEEE Access*, *5*, 25555–25564. https://doi.org/10.1109/ACCESS.2017.2771807

Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*. http://arxiv.org/abs/1412.6980

Lee, S., Kim, J., Shin, S., Porras, P., & Yegneswaran, V. (2017). Athena: A Framework for Scalable Anomaly Detection in Software-Defined Networks. *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 249–260. https://doi.org/10.1109/DSN.2017.42

Lu, M., Deng, W., & Shi, Y. (2016). TF-IdleTimeout: Improving efficiency of TCAM in SDN by dynamically adjusting flow entry lifecycle. *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 002681–002686. https://doi.org/10.1109/SMC.2016.7844645

Luo, H., Li, W., Qian, Y., & Dou, L. (2018). Mitigating SDN Flow Table Overflow. *2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC)*, *19*(6), 821–822. https://doi.org/10.1109/COMPSAC.2018.00137

Nguyen, T. N. (2018). The Challenges in SDN/ML Based Network Security : A Survey. *IEEE Internet of Things Journal*, *5*(6), 4829–4842. https://doi.org/10.1109/CSNET.2018.8602680

Polat, H., Polat, O., & Cetin, A. (2020). Detecting DDoS Attacks in Software-Defined Networks Through Feature Selection Methods and Machine Learning Models. *Sustainability*, *12*(3), 1035. https://doi.org/10.3390/su12031035

Song, C., Park, Y., Golani, K., Kim, Y., Bhatt, K., & Goswami, K. (2017). Machine-Learning Based Threat-Aware System in Software Defined Networks. *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, 1–9. https://doi.org/10.1109/ICCCN.2017.8038436

Sultana, N., Chilamkurti, N., Peng, W., & Alhadad, R. (2019). Survey on SDN based network intrusion detection system using machine learning approaches. *Peer-to-Peer Networking and Applications*, *12*(2), 493–501. https://doi.org/10.1007/s12083-017-0630-0

Tang, T. A., Mhamdi, L., McLernon, D., Zaidi, S. A. R., & Ghogho, M. (2018). Deep Recurrent Neural Network for Intrusion Detection in SDN-based Networks. *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, 202–206. https://doi.org/10.1109/NETSOFT.2018.8460090

Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 1–6. https://doi.org/10.1109/CISDA.2009.5356528

Yan, B., Xu, Y., & Chao, H. J. (2018). Adaptive Wildcard Rule Cache Management for Software-Defined Networks. *IEEE/ACM Transactions on Networking*, *26*(2), 962–975. https://doi.org/10.1109/TNET.2018.2815983

Yang, B., Wu, Y., Chu, X., & Song, G. (2016). Seamless Handover in Software-Defined Satellite Networking. *IEEE Communications Letters*, *20*(9), 1768–1771. https://doi.org/10.1109/LCOMM.2016.2585482

Zahid, M. S. M., Isyaku, B., & Fadzil, F. A. (2017). Recovery of Software Defined Network from Multiple Failures: Openstate Vs Openflow. *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, 1178–1183. https://doi.org/10.1109/AICCSA.2017.32

Zhang, H., Cai, Z., Liu, Q., Xiao, Q., Li, Y., & Cheang, C. F. (2018). A Survey on Security-Aware Measurement in SDN. *Security and Communication Networks*, *2018*. https://doi.org/10.1155/2018/2459154

Zhou, Y., Chen, K., Zhang, J., Leng, J., & Tang, Y. (2018). Exploiting the Vulnerability of Flow Table Overflow in Software-Defined Network: Attack Model, Evaluation, and Defense. *Security and Communication Networks*, *2018*, 1–15. https://doi.org/10.1155/2018/4760632