LSTM-NB: DoS Attack Detection On SDN With P4 Programmable Dataplane

***Abstract*—This paper proposes LSTM-NB, a combination of Long Short-Term Memory (LSTM) and Naive Bayes (NB) algorithms to tackle Denial of Service (DoS) attacks on Programming Protocol-independent Packet Processors (P4) language-based Software Defined Network (SDN). The implementation of SDN is becoming more popular. However, there are critical aspects of the SDN architecture, one of which is that it is vulnerable to DoS attacks that can cause the network to lose the availability principle of the CIA Triangle. There are a number of works have been proposed to overcome this vulnerability, however, the threat is still exist. The proposed technique achieves an accuracy of 88% on SDN-DL Dataset, 98% on NSL-KDD, and 96% on CICIDS2017 with FNR score between 1-2%. In addition, we compare our proposed technique with other machine-learning and deep-learning methods. Through extensive experimental evaluation, we conclude that our proposed approach exhibits a strong potential for DoS detection in the SDN environments.**

***Index Terms*—Computer Network Security, Intrusion Detection System (IDS), Machine Learning, Deep Learning, Denial of Service (DoS).**

I. Introduction

Computer networking is a complex matter and difficult to manage. It happens due to the large number of equipment used on the network [1]. Besides, the devices on traditional networks have designs, software, and hardware related to one vendor. Each other vendor has different designs and devices [2]. To tackle those complexities, Software Defined Network (SDN) technique [1] is proposed. SDN has different characteristics when it is compared to traditional networks which is the separation of the control plane and data plane of a network device [1], [2]. SDN applies the concept of centralization to its network architecture like a traditional network. However, it makes SDN architecture network vulnerable to cyber-attacks [3]. There are three types of attacks targeting SDN networks. These are fraud attacks, intrusion attacks, and malicious tampering attacks [4]. One of the attacks on the SDN network was Denial of Services; this vulnerability was caused by the architecture of SDN [5].

Denial of Services, commonly known as DoS, is a cybercrime with the method of sending packets excessively and aiming to exploit the resources of the network [6]. The separation of the control and data planes causes DoS attack in SDN [5]. Attackers exploit both the control plane and data plane [7], so it will disrupt the flow rule decision and can also result in the occurrence of a bottleneck on the network. It can be harmful if there is a failure on the network component [8]. There are two types of DoS attacks, namely volumetric attacks such as ICMP-Flood, UDP-Flood, and TCP-SYN Flood, and application-layer attacks [8]–[10]. We proposed LSTM-NB, a combination of Long Short-Term Memory (LSTM) and Naive Bayes (NB) algorithms, to solve these problems.

A. Related Work

An intrusion Detection System (IDS) is used to prevent DoS attacks. IDS inspects every activity that occurs on the network [6], [11], [12]. There are two types of IDS namely **signature-based** and **anomaly-based**. Both approaches have drawbacks, such as low intelligence and weak adaptability if applied traditionally. Hence, it is ineffective when implemented in many scenarios [13].

To overcome those issues, we need a dynamic approach. During the last decade, there have been many surveys and reviews of the technology used in IDS, one of which is technology by applying the machine learning method [3], [13]. Machine-learning methods commonly used are SVM, Random Forest, KNN, and technologies such as Artificial Neural Network [13]. On IDS implementation, most of the methods are currently still using variations of Shallow Learning where it needs continuous learning on the model to update its capabilities. Besides, it needs more in-depth analysis to select the features used [5]. The next drawback is that the system can only detect some DoS attacks [3], [13]. Deep Learning methods can be used to solve this problem [3], [5]. Deep Learning was chosen to solve the problem because of its learning ability, and generalization of the existing attributes [5].

B. Paper contribution and organization

This paper proposes LSTM-NB to make DoS early detection system. The proposed technique is inspired by a research conducted by Ahuja et al. which combined two different deep learning algorithms (CNN-LSTM). However, the CNN-LSTM method needs extensive computation resources. Moreover, as the CNN-LSTM is a deep learning technique, it skips the feature selection stage and neglects a flexible and configurable system. Due to this reasons, we propose LSTM-NB where we implement deep learning and shallow learning. We improve shallow learning performance by using deep learning but consume fewer computation resources.

Recurrent Neural Network (RNN) has shown great success in language modelling, text generating, and speech recognition; based on Tang et al., RNN is believed to be a powerful technique to represent the relationship between current and past events and enhance anomaly detection system [2]. The RNN-based algorithm was chosen because of its advantages

in handling time-series data. Besides that, Musumeci et al., in their research, advised implementing the RNN algorithm for DoS Detection. However, RNN has some disadvantages; one of the problems is the Long Dependency Problem; in theory, RNN can solve that problem, but in practice, RNN cannot solve it. This problem is called the Vanishing gradient problem.

LSTM was chosen because of its advantages and can optimize the long dependency problem in the Recurrent Neural Network. Besides that, LSTM also can keep records of information on packets that have passed the system built [14] so that the packet analysis is expected to be more accurate with this method. Naive Bayes was chosen because it is pretty simple to implement and has high accuracy [14].

The main contributions in this paper are as follows:

- We compare different ML algorithms to detect DoS attacks such as Naive Bayes, LSTM, ANN, and LSTM-NB in terms of accuracy, recall, precision, and false-negative rates.
- We compare the performance of the machine-learning model on NSL-KDD, SDN-DL, and simulation-generated datasets.
- We provide P4-based data plane code for simulation, implementing deep learning intrusion detection system, and simulation features extraction code.

The present paper is organized as follows. In Sec. II, we provide background on Software Defined Network, Denial of Services attacks and P4 language. Then in Sec. III, we overview the proposed system schema. After that, in Sec. IV We describe the experimental framework, such as evaluation methods, dataset, and data preprocessing. In Sec V, we provide discussions of the experimental results. Finally in Sec VI, we provide the conclusion.

II. Background

A. DoS Attack

Denial of Service is one of the cybersecurity attacks that aim computer networks and make computer networks inaccessible [15]. This attack targets communication nodes such as network infrastructure or components. The methods is used by flooding the network with packets and making the network low on resources (overwhelm) make the network offline for some time. Currently, there are several types of prevalent DoS attacks, namely UDP Flood, ICMP Flood, TCP Flood, HTTP Flood, and HTTPS Flood [16]. We can group them based on their protocol like UDP, TCP, HTTP, and ICMP attack. The attack can easily be classified based on the methods used. DoS attacks have wide varieties and methods used. However, they usually have similarities that is overwhelming the network using packet [6]. According to Kaspersky Lab data, since the beginning of the COVID-19 pandemic, DoS attack rate increases up to 20% since online activities growth. The attack mainly consists of three types of attack SYN, UDP, and TCP attack, where SYN attacks are 78.20%, followed by UDP 15.17%, and followed by TCP attacks as much as 5.67%. To minimize DoS attack impact, it can be avoided by making an early detection system.

B. Software Defined Network

Software-Defined Network (SDN) is an innovation in networking that changes how we design and manage the network itself, using SDN management, control, and creating innovation that is easier and possible to implement [1]. SDN provides a new paradigm option that implements a centralized system; it separates two parts of the network. The control plane tasked to control the network, and the data plane in charge of forwarding packet [1], [17], [18]. In addition, traditional networks which are closed, proprietary to the control and have differences between vendors, making it for network administrators hard to manage and configure [1], [2]. SDN builds on the Application, Control, and Infrastructure layers. This architecture allows SDN to implement the centralized architecture and then have the advantage of modifying the network. Furthermore, infrastructure and all these layers are connected using southbound and northbound API to communicate with each other. Currently, the well-known SDN protocol is Openflow, which provides a simple and robust SDN system. However, OpenFlow has disadvantage which is lack programmability. To tackle this problem, we use a more powerful solution that is P4-language Dataplane.

C. P4 Language

P4 language is a high-level programming language for routers and switches, designed to allow programming on data plane components such as hardware or software switches, network interface cards, routers, etc [3]. P4 is an open-source language while OpenFlow P4 uses top-Down design. The main difference between P4 and OpenFlow is its programmability. In this paper, the simulation uses the behavioural model using the Mininet simulator to simulate a DoS attack. Based on Musumeci et al. a P4 program composed of the following component:

- *Parsers* had a function to identify the allowed protocols and fields in the program. Typically, they contain the names of the used headers and their size in bits.
- *Control Plane (Ingress/Egress)* had a function to describe the order of processing rules that is applied to the packet.
- *Table* had a bunch of processing rules that form ”match-action”. When the P4 program processes packets, the ingress pipeline is executed to look for the matching rule(s) which fit the incoming packet.

III. Proposed Detection System

This section discusses the proposed system and describes the system overview and methods used in this paper.

A. System Overview

The system is built with two main components. First, the data collector module and second, the detection system

governed by three main modules. *Preprocessing*, this module prepares data from the data collector matching with classification. Module input type, this module contains feature selection module. The last module is normalization which uses Min-Max Scaler module to Normalize data after the data is passed to LSTM Module.

1) LSTM Module

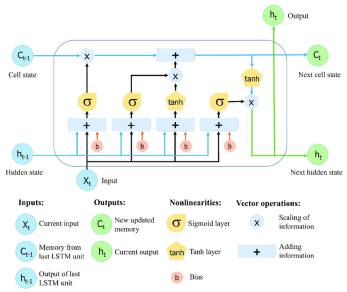

Fig. 1. LSTM Architecture [19]

LSTM Module contains LSTM-classifier. LSTM is an application of the Recurrent Neural Network which has the ability to learn about long-standing dependencies [5], where the structure of the LSTM cell itself is depicted as shown in Figure 1. In Figure 1, every t time of this LSTM cell is controlled by various logic gates, which aim to maintain or reset the values in the cell. Figure 1 shows three types of gates, the gates are located sequentially from the left side, the gate consists of *Forget gate* (f_t), *Input gate* (i_t) , and *Output gate* (o_t) all of which have sigmoid activation functions. Hence, there is one gate that uses the tanh function called *candidate value gate*.

$$c_t = f_t \odot c_{t-1} + i_t \odot c_t^t \tag{1}$$

$$h_t = o_t \odot \tanh(c_{t-1}) \tag{2}$$

First step on LSTM is the forget gate to decide how much value is removed from the state of the cell, and next stage input gate determines how much new information stored, for next step (c_t) is described based on equation and for LSTM result (h_t) is defined by equation.

2) Naive Bayes Module

Naive Module contains Naive Bayes Classifier. Naive Bayes is a method for classifying based on Naive Bayes theory [20]. This method is a classification using a simple probabilistic approach. We need to calculate a set of probabilities by adding up the frequency and combination of values from the data.

$$P(H|E) = \frac{P(H|E)P(H)}{P(E))} \tag{3}$$

$$P(H|E) = P(E1|H) \times P(E2|H) \times \cdots \times P(En|H) \times P(H) \tag{4}$$

Values can be assigned to these attributes and the patterns resulting from these calculations is used for classification. The Naive Bayes system itself is classified as *supervised learning* in the application of *machine-learning* [20]. It defined as equation 3 and equation 4. In Naive Bayes, the calculation of the probability of an event H which is a condition in data E is carried out by first calculating the probability from data E with condition H **(P(H|E))**, after that, the result **(P(H|E))** is multiplied by the probability condition H **(P(H))**, then divided by the probability data E (P(E)). Hence, the detection of DoS attacks can be carried out using a calculation of the probability of the attack occurring on the data then divided by the probability of data E [20].

IV. Experimental Framework

This section describes the datasets and evaluation metrics used in the experimental framework. We also describe the data preprocessing procedure.

A. Dataset

The experimental evaluation framework use widely popular datasets which is NSL-KDD and CICIDS 2017 [2]. NSL-KDD datasets are one of the most popular datasets used on NIDS Performance. These datasets were introduced by Tavallace et al., but these datasets are out of date and lack traffic diversity and feature sets. CICIDS2017 is used to solve that problem because the datasets are relatively new. These two datasets are not specified for SDN architecture. However, the datasets are used as there is lack of public datasets for DoS attacks on SDN architecture. Several researchers generated many SDN datasets manually, but it is closed and hence researchers still use conventional datasets to evaluate their model [2]. In this paper, in addition to NSL-KDD and CICIDS 2017, we also use a dataset related to SDN architecture provided by Ahuja et al., to develop a deep learning anomaly detection system on an SDN-based network [21]. This dataset is used to prove that our proposed system works on SDN architecture and significantly improve the performance. Lastly, the simulation dataset is created using SDN simulation using Iperf3 and Hping3 to generate network traffic and then captured by a network sniffing application.

B. Data Preprocessing

On NSL-KDD and CICIDS 2017, we performed one-hot encoding, scaling, and label transformation for features needed. We did not do a feature selection process for this dataset when inputted to the LSTM module, but we selected some features, mainly LSTM prediction result and protocol type feature. For the SDN-DL dataset, we performed feature selection using Heatmap, Chi-Square, Tree Classifier, and Data Slice. Besides that, we performed missing value and duplicate value handling. Lastly, we performed normalization using min-max methods. We did label encoding for some features like protocol type, and the last dataset we used is the simulation dataset we generated before we performed data normalization and feature selection because this data was used to train our model to simulation data. .

C. Evaluation Methods

This paper uses parameters such as Accuracy, Precision and False Negative Rate to calculate detection system performance. To produce that, we need to make a confusion matrix. Based on Qin et al. confusion matrix contains the item defined below.

- **True Positive (TP)** is parameter of DoS Packet classified as DoS Condition.
- **True Negative (TN)** is parameter of Normal Packet classified as Normal Condition.
- **False Positive (FP)** is parameter of Normal Packet classified as DoS Condition.
- **False Negative (FN)** is parameter of DoS Packet classified as Normal Condition..

Based on the parameter explained before, we calculate performance parameters such as Accuracy, Precision, and False Negative Rate. We use an equation based on Li et al. and Aljarwaneh et al. research to calculate the performance parameter.

- ***Accuracy*** compares the correct classification with the total number in the dataset.

$$Acc = \frac{TP+TN}{TP+TN+FP+FN} \tag{5}$$

- ***Precision*** is part of the data that is classified as positive and has a true positive value.

$$Precision = \frac{TP}{TP+FP} \tag{6}$$

- ***False Negative Rate*** is a comparison that shows the number of incorrect packets that are classified as true.

$$FNR = \frac{FN}{TP+FP} \tag{7}$$

From Equation 5 - 7, they consist four values: TN (True Negative), TP (True Positive), FN (False Negative), and FP (False Negative).

V. Experiment and Result

In this section, we describe three evaluations and discuss the results. Evaluation 1 uses SDN-DL dataset, and then we compare the proposed method using some Vanilla algorithms on Sklearn and Tensorflow library, such as ANN, Naive Bayes, and LSTM. After that, all methods is evaluated using evaluation parameters. Evaluation 2 uses NSL-KDD and CICIDS datasets and it is compared to Tang et al. research. In Evaluation 3, we evaluate the performance of the P4-based SDN network using the dataset we generated before hand and present the result.

A. Evaluation 1 - SDN-DL Dataset Evaluation

The goal of evaluation 1 is to carry out comparative analysis of proposed system with the previous research using the same dataset. This evaluation is focused to find features that have optimal performance. On this evaluation we perform data exploration, cleansing, transform, and normalize data using Min-Max Scaling, after that we split data to three parts: training, validation, and testing, which contains 70% data for training from training split we split again 70% data for training and 30% data for validation, and lastly 30% for testing. Data exploration on SDN-DL Dataset finds no missing and duplicated values.

TABLE I
Feature Selection Result

Algorithm	Accuracy (%)		Precision (%)		FNR (%)	
	Train	Test	Train	Test	Train	Test
ANN (Full Feature)	**92.96**	**92.85**	**93**	**92**	**14.5**	**14.6**
ANN (Chi Square)	69.2	69.29	78	78	43.9	43.9
ANN (Extra Tress)	93.81	93.87	94	94	15.6	15.2
ANN (Heatmap Slice)	84.99	85.10	84	84	10.9	10.8
LSTM (Full Feature)	**94.24**	**94.09**	**94**	**94**	**14.2**	**14.2**
LSTM (Chi Square)	80.78	80.39	81	81	16.6	16.6
LSTM (Extra Tress)	90.79	90.78	90	90	13.4	13.4
LSTM (Heatmap Slice)	88.57	87.86	88	87	12.6	12.5
NB (Full Feature)	60.79	61.08	30	31	-	-
NB (Chi Square)	60.79	61.08	30	31	-	-
NB (Extra Tress)	**61.16**	**60.51**	**31**	**30**	-	-
NB (Heatmap Slice)	61.01	60.78	31	30	-	-

Next step we conducted feature selection. We select feature using four ways. The first way we use all features, second we apply chi square calculation to get feature importance score. The results are *'dst','src','Protocol','pktcount','pktcount'*. The third way we use extratreesclassifier to calculate important methods. From this methods we obtain *'pktrate','pktperflow','Protocol', 'src','dst'*. The last method used is to check the correlation heatmap and combined with another selected feature. We decide to choose *'dst','src','Protocol','bytecount'*. After the features were selected, then we perform normalization process using two methods: Normalization, and Standarization methods. However, after performing an evaluation, several algorithms cant handle standarization value. Hence, we choose Normalization for feature selection evaluation. We evaluate feature selection using LSTM, VanillaNB, and ANN methods. The results are presented on Table I.

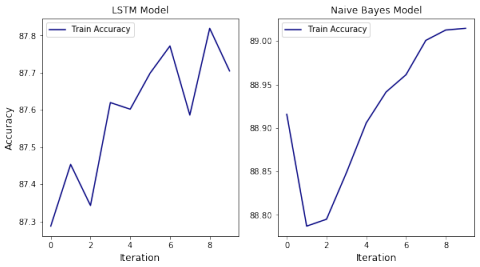

Fig. 2. LSTM and Naive Bayes Model Accuracy on Training Session

From Table I, it can be seen that Deep-Learning Based methods such as LSTM and ANN are preferable to use all features. However, the shallow learning algorithm, which is Naive Bayes, requires feature selection process. Moreover, Naive Bayes method has lousy performance as it only detects non-anomalies value. From this result, it can be identified that heatmap method has the lowest performance. Therefore, we decide to improve the performance using our proposed

method. When developing the proposed method, we use 10 epoch to train LSTM model and using Cross-Validation to find optimum var_smoothing value. We apply 10 fold and 1000 tries on Naive Bayes model and get the optimum value is 0.732596542821523. The result obtains accuracy at 89%. The achieved accuracy can be seen on Figure 2. Moreover, we improve the Naive Bayes Slice method performance from 61% to 89%. The we conduct test and compare the result with the previous work as shown on Table II - IV.

TABLE II
ACCURACY PERFORMANCE COMPARISON

Algorithm	Accuracy (%)		
	Training	Validation	Testing
LSTM + NB (Combined)	88.77	88.18	88.02
LSTM	88.59	87.89	88.57
Naïve Bayes	62.01	61.46	61.85
ANN	87.698	87.67	87.69
CNN-LSTM [21]	**99.48**		

TABLE III
PRECISION PERFORMANCE COMPARISON

Algorithm	Precision (%)		
	Training	Validation	Testing
LSTM + NB (Combined)	85.21	82.41	84.16
LSTM	86.02	85.03	83.14
Naïve Bayes	51.35	51.20	51.30
ANN	83.29	83.51	83.36
CNN-LSTM [21]	**99.55**		

TABLE IV
FNR PERFORMANCE COMPARISON

Algorithm	FNR (%)		
	Training	Validation	Testing
LSTM + NB (Combined)	13.60	10.78	14.53
LSTM	15.72	16.67	10.58
Naïve Bayes	64.16	62.26	64.47
ANN	14.09	14.09	14.09
CNN-LSTM [21]	**3**		

From Table II - IV, it can be seen that our proposed system obtains a lower performance compare to Ahuja et al. However, their result is expected as they used a fully Deep Learning method. The drawback is that Deep Learning consumes more resource and has no feature knowledge to configure the network. However, if we compared to vanilla machine-learning and deep learning algorithm, ours has the highest accuracy, precision, and lowest FNR rate.

B. Evaluation 2- NSL-KDD and CICIDS Dataset Evaluation

The goal of evaluation 2 is to carry out a comparative analysis with research conducted by Tang et al. The main reason this evaluation was conducted was to evaluate the proposed system with a public network dataset. In addition, we need to compare the LSTM-based model with the RNN-based model. We split the data into three parts: Training, Validation, and Testing, which contains 70%:30% data partition on Training and Testing. After that, we conduct data preprocessing, and then we use all attribute to train our LSTM-NB model. The performance parameter we used to make the comparison is accuracy and precision. FNR was not used because research before did not describe that value. Results of this evaluation are shown in Table V.

TABLE V
NSL-KDD AND CICIDS2017 PERFORMANCE COMPARISON

Algorithm	NSL-KDD		CICIDS2017	
	Accuracy	Precision	Accuracy	Precision
LSTM + NB (Combined)	98.85	99.03	96.4	97.8
GRU-RNN [2]	89	91	99	99
DNN [2]	75.9	0	75.75	0
SVM [2]	65.67	0	69.52	0
NBTree [2]	-	-	82.02	0
VanillaNB [2]	44.39	0	-	-

From Table V it can be seen that our proposed system can achieve an accuracy of 99.85% on the NSL-KDD dataset and 96.4% on CICIDS2017. It means that our proposed system works significantly better than DNN, SVM, VanillaNB, and Tang et al. proposed system GRU-RNN on NSL-KDD dataset. Besides that, on CICIDS 2017, our proposed system achieved slightly lower than the GRU-RNN dataset, meaning the proposed system worked significantly accurate and had a good performance. To achieve this value, we conduct data transformation using one-hot encoding and data preparation; after that, we split data 70%:30%. We developed with the same methods as evaluation one, but the difference is that we did not choose any feature for the LSTM module, but we selected 'lstm_result_1', 'lstm_result_2', 'lstm_result_3', and protocol name on Naive Bayes module. We developed the Naive Bayes module using Grid Cross-Validation methods to get the optimum var_smoothing value, and the value used is 0.732596542821523. Besides that, on this evaluation, we got an FNR rate range between 1-2% and this value was lower than previously proposed methods CNN-LSTM.

C. Evaluation 3- Simulation Dataset Evaluation

The goal of evaluation 3 is to identify which features can be used on a P4-based SDN Network as P4 has some different features that can be extracted with OpenFlow SDN. We extracted 'src'. 'dst'. 'length', and 'protocol'. We use Accuracy and FNR to evaluate our model performance with P4-Mininet Simulation. This simulation is generated using Hping3 and Iperf. We conduct 30 minutes network simulation, and after that, we save data based on the session we conduct. In the first session, we conduct normal traffic with Iperf3 and variate windows and packet lenght between 8 -2.4 Kbits, and in another session, we conduct an attack session we conduct UDP, SYN, TCP Flood, after that we preprocessed data, and use proposed system. The performance we got is accuracy 100% with FNR 0%, which can happen because the variation

of the DoS attack is obsolete. However, this proven system can provide the classification of DoS attacks.

VI. Conclusions

From the results, it can be justified that our proposed method gives acceptable performance and less complexity in contrast to a fully deep learning technique. The proposed method achieves 88% accuracy, 85% precision, and 10% FNR. Using NSL-KDD and CICIDS, our proposed schema achieved $\geq$ 96% accuracy. The data obtain from simulation using P4-Mininet achieved 100% accuracy. It proves that our proposed LSTM-NB can be implemented with acceptable performance on SDN environment.

References

[1] N. Feamster, J. Rexford, and E. Zegura, "The road to sdn: an intellectual history of programmable networks," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 87–98, 2014.

[2] T. A. Tang, D. McLernon, L. Mhamdi, S. A. R. Zaidi, and M. Ghogho, "Intrusion detection in sdn-based networks: Deep recurrent neural network approach," in *Deep Learning Applications for Cyber Security*. Springer, 2019, pp. 175–195.

[3] F. Musumeci, V. Ionata, F. Paolucci, F. Cugini, and M. Tornatore, "Machine-learning-assisted ddos attack detection with p4 language," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[4] Y. Qin, J. Wei, and W. Yang, "Deep learning based anomaly detection scheme in software-defined networking," in *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*. IEEE, 2019, pp. 1–4.

[5] M. P. Novaes, L. F. Carvalho, J. Lloret, and M. L. Proença, "Long short-term memory and fuzzy logic for anomaly detection and mitigation in software-defined network environment," *IEEE Access*, vol. 8, pp. 83 765–83 781, 2020.

[6] N. Moustafa, J. Hu, and J. Slay, "A holistic review of network anomaly detection systems: A comprehensive survey," *Journal of Network and Computer Applications*, vol. 128, pp. 33–55, 2019.

[7] A. Sangodoyin, T. Sigwele, P. Pillai, Y. F. Hu, I. Awan, and J. Disso, "Dos attack impact assessment on software defined networks," in *International Conference on Wireless and Satellite Systems*. Springer, 2017, pp. 11–22.

[8] M. S. Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Ddosnet: A deep-learning model for detecting network attacks," in *2020 IEEE 21st International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*. IEEE, 2020, pp. 391–396.

[9] S. R. Talpur and T. Kechadi, "A survey on ddos attacks: Router-based threats and defense mechanism in real-world data centers," in *2016 Future Technologies Conference (FTC)*. IEEE, 2016, pp. 978–984.

[10] O. Yevsieieva and S. M. Helalat, "Analysis of the impact of the slow http dos and ddos attacks on the cloud environment," in *2017 4th International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T)*. IEEE, 2017, pp. 519–523.

[11] S. Anwar, J. Mohamad Zain, M. F. Zolkipli, Z. Inayat, S. Khan, B. Anthony, and V. Chang, "From intrusion detection to an intrusion response system: fundamentals, requirements, and future directions," *Algorithms*, vol. 10, no. 2, p. 39, 2017.

[12] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in internet of things," *Journal of Network and Computer Applications*, vol. 84, pp. 25–37, 2017.

[13] M. Zhang, J. Guo, B. Xu, and J. Gong, "Detecting network intrusion using probabilistic neural network," in *2015 11th International Conference on Natural Computation (ICNC)*. IEEE, 2015, pp. 1151–1158.

[14] Y. Li and Y. Lu, "Lstm-ba: Ddos detection approach combining lstm and bayes," in *2019 Seventh International Conference on Advanced Cloud and Big Data (CBD)*. IEEE, 2019, pp. 180–185.

[15] V. Zlomislić, K. Fertalj, and V. Sruk, "Denial of service attacks, defences and research challenges," *Cluster Computing*, vol. 20, no. 1, pp. 661–671, 2017.

[16] M. Aamir and M. Arif, "Study and performance evaluation on recent ddos trends of attack & defense," *International Journal of Information Technology and Computer Science*, vol. 5, no. 8, pp. 54–65, 2013.

[17] H. Polat, O. Polat, and A. Cetin, "Detecting ddos attacks in software-defined networks through feature selection methods and machine learning models," *Sustainability*, vol. 12, no. 3, p. 1035, 2020.

[18] S. Azodolmolky, *Software defined networking with OpenFlow*. Packt Publishing, 2013, vol. 153.

[19] X.-H. Le, H. V. Ho, G. Lee, and S. Jung, "Application of long short-term memory (lstm) neural network for flood forecasting," *Water*, vol. 11, no. 7, p. 1387, 2019.

[20] A. Mehmood, M. Mukherjee, S. H. Ahmed, H. Song, and K. M. Malik, "Nbc-maids: Naïve bayesian classification technique in multi-agent system-enriched ids for securing iot against ddos attacks," *The Journal of Supercomputing*, vol. 74, no. 10, pp. 5156–5170, 2018.

[21] N. Ahuja, G. Singal, and D. Mukhopadhyay, "Dlsdn: Deep learning for ddos attack detection in software defined networking," in *2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. IEEE, 2021, pp. 683–688.