A new approach of extremely randomized trees for attacks detection in software defined network

Hasan Kamel, **Mahmood Zaki Abdullah**
Department of Computer Engineering, College of Engineering, Mustansiriyah University, Baghdad, Iraq

Article Info

Article history:

Received May 21, 2022
Revised Aug 12, 2022
Accepted Sep 7, 2022

Keywords:

Distributed denial of service attacks
Extremely randomized trees
Machine learning
Particle swarm optimization
Software defined networking

ABSTRACT

Software defined networking (SDN) is the networking model which has completely changed the network through attempting to make devices of network programmable. SDN enables network engineers to manage networks more quickly, control networks from a centralized location, detect abnormal traffic, and distinguish link failures in efficient way. Aside from the flexibility introduced by SDN, also it is prone to attacks like distributed denial of service attacks (DDoS), that could bring the entire network to a halt. To reduce this threat, the paper introduces machine learning model to distinguish legitimate traffic from DDoS traffic. After preprocessing phase to dataset, the traffic is classified into one of the classes. We achieved an accuracy score of 99.95% by employing an optimized extremely randomized trees (ERT) classifier, as described in the paper. As a result, the goal of traffic flow classification using machine learning techniques was achieved.

Corresponding Author:

Hasan Kamel
Department of Computer Engineering, College of Engineering, Mustansiriyah University
Al-Bab Al-Muadham Street, Baghdad, Iraq
Email: egma003@uomustansiriyah.edu.iq

1. INTRODUCTION

Software defined networkings (SDN's) aim is to centralize network topology. In contrast to traditional networks, which have two control and data layers that are linked together, SDN has just one control level (SDN controller) for all devices [1]. In SDN, data transmission is managed by software that controls traffic between devices. This is in contrast to current network infrastructure, in which network traffic controlled by hardware and described by routers, switches, and another network equipments. The control level is transferred to another element called the controller in the centralized SDN network, and the SDN switch is only made up of the data level [2], [3]. The controller serves as the network's brain, and it is a network-wide centralized element, with all devices in network adhering to the controller's decisions [4]. The control level is in control of traffic routing, while the data level is in control of simple data packet forwarding [5].

On SDN, many distinct types of attacks are possible, including denial of service (DoS), traffic diversion, application programming interface (API) exploitation, and address resolution protocol (ARP) spoofing [6]. The type of attack we're dealing with here is a distributed denial of service attack (DDoS) [7]. DDoS is a more advanced type of DoS attack. DoS is a threat in which the purpose is to make a device unavailable to users by interfering with the host's services, which is fulfilled by flooding the machine with transactions and overloading the platform. In DDoS, requests are being sent to the several different sources, making it impossible to stop the attack, in contrast to when only one host is attacking since one host can be inhibited, but many sources are hard to determine and block [8]. In our work, we used a customized dataset ("DDOS attack SDN Dataset") extracted from a software defined network (SDN) environment and provided by leadingindia.ai [9]. This attack was used because this attack is considered one of the dangerous attacks on any network and it is still working. The attacker's goal in DDOS attack [10], flood the target host's resources in order to disrupt the benign host. As a result, we made an effort in our work to apply machine learning techniques to detect DDOS attacks. The various attack classes represented in the dataset are as follows: (TCP-SYN, UDP-flooding and ICMP-flooding attacks). The structure of paper in this manner: section 2 includes the literature review, while section 3 depicts the proposed method, section 4 depicts the results and discussion, and section 5 depicts the conclusion.

2. LITERATURE REVIEW

One of the majority crucial aspects of network administrators is traffic classification. In general, there's many aspects to traffic flow classification. Deep-packet-inspection yields good results, and it can only use in an unencrypted traffic. Whereas the majority of data is encrypted in the real world, port based approach that seems to be easy and rapid but easily duped and thus untrustworthy, finally there is an approach using artificial intelligence. Later is regarded as trustworthy, and it will be the primary goal of our task for the remnant of this work (paper). Karan *et al.* [11] presented a DDoS assault detection mechanism for SDN. For attack classification, they used the support vector machine (SVM) classifier model and deep neural networks (DNN) machine learning model. The results demonstrate that DNN has a higher accuracy rate than SVM, with a rate of 92.30. Nam *et al.* [12] suggested a DDoS safety system that detects attack flows using the SDN architecture. Their hybrid approach employs a combination of K-nearest neighbor (K-NN) and self-organizing map (SOM) techniques. Using flow statistical data obtained from SDN, they categorized the traffic as regular or malicious and they achieved an accuracy of 98.24%. Adhikary *et al.* [13] concentrated on a hybrid approach that merged decision trees (DT) and neural network techniques for distinct kinds of DDoS-attacks in vehicular ad hoc networks (VANET) system. The suggested hybrid algorithm outperforms the single models of DT and neural networks.

The authors concentrated on detecting DDoS attacks on the (IoT) system. Ujjan *et al.* [14] collect network incoming traffic to SDN data level, their proposed approaches used time based and packet based activities planned. They hope to reduce the computational of the trusion detection systems (IDS) and DNN while improving classification efficiency by these sampling-approaches. According to the outcomes, their suggested framework does have higher detection results. Nanda *et al.* [15] used ML algorithms to create a model. The model was trained through using knowledge gleaned from previous attacks or experiences and NSL-KDD to define malicious attacks and connections. To propose the model, the most commonly used ML techniques are Naïve Bayes., C4.5, decision table and Bayesian network. The accuracy of prediction of the Bayesian network is greater than the other models, which is 91.68%. Sahoo *et al.* [16] proposed a modern evolutionary model for categorizing DDoS assault in SDN network. For malicious data classification, the model employs a combined SVM model, and genetic algorithms (GA) were used for SVM enhancement (optimization). According to the experiments, the suggested combined method has an accuracy of 98.9%. Kyaw *et al.* [17] detected user data protocol (UDP) intrusions in the SDN network using two ML algorithms. For traffic packet creation, Scapy tool was employed. The flow statics are collected by their system through the open flow switch.

After feature extraction stage, they discussed the classification results of linear and polynomial SVM models. According to test findings, the Polynomial SVM has an accuracy of 95%. Tonkal *et al.* [18] selecting the most pertinent features by using the NCA technique, they perform an efficient classification. The exploratory findings demonstrate that DT algorithm achieves 100% prediction performance. Cil *et al.* [19] used a dataset called CICDDoS2019 that contained the renowned DDoS attack kinds created in 2019. They discovered that threats on network activity were identified with 99.99 achievement and threats kinds were labeled with an overall accuracy of 94.57. Obviously, they got good results but the dataset they used is extracted from traditional networks.

Literature review show that many researchers have used many machine learning techniques as well as different datasets to classify traffic. But it is clear that there is a need to improve the accuracy and efficiency of machine learning models. In addition, due to the growing sophistication of attack techniques, it is obligatory to use a fresh dataset extracted from the SDN environment. So, in this study we used the recent dataset ("DDOS attack SDN Dataset") [9]. We used ERT as a classifier to get better results, however ERT have some limitations. One such limitation is that the classifier's performance is contingent to the selection of appropriate hyperparameters. So, we used particle swarm optimization (PSO) to optimize the ERT hyperparameters.

3. PROPOSED METHOD

To improve DDos attacks classification in SDN, we propose an optimized extremely randomized trees model, in which each case in a dataset is labeled as "attack" or "normal," and the suggested algorithm is trained on the class data. Figure 1 demonstrates the proposed model's structure. Which is divided into various stages.

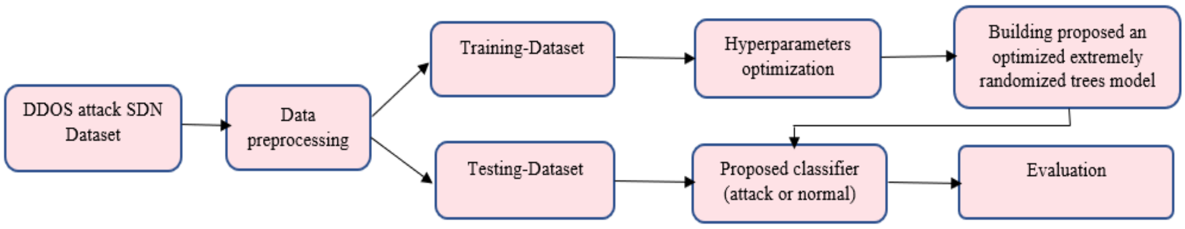
DDOS attack SDN Dataset
Data preprocessing
Training-Dataset
Testing-Dataset
Hyperparameters optimization
Building proposed an optimized extremely randomized trees model
Proposed classifier (attack or normal)
Evaluation

Figure 1. Block diagram of the proposed method

3.1. Dataset preprocessing

The dataset ("DDOS Attack SDN Dataset") is newly constructed using the SDN framework and contains 104 thousand records and 23 features and can be found in Mendeley-data [8]. The traffic flows in this dataset use three distinct protocols: ICMP, TCP and UDP. The label indicates if the traffic is malicious or not. The data contains characteristics that define the destination and source, as well as knowledge about transfer duration, messages, bytes, data rate, and so on. Several preprocessing techniques were applied to the data set. Missing value handling, null value removal, categorical value encoding, and other pre-processing techniques are used. The next step was to determine whether the variables were numerical or categorical, and then encode categorical variables. The categorical variables in this case were protocol, destination, switch, and source. Only those categorical data [20] were encoded using a onehot-encoding method. Then, using a correlation and heatmap plot techniques, we attempted to find the correlation (connection) between the input and output features. The column containing time information and displayed with the "dt" feature was determined to be unnecessary and was removed from the dataset. Finally, the numeric features were normalized [21] and pre-processing was completed.

3.2. Hyperparameters optimization based on optimization algorithm

In this part we introduce an explanation of the suggested approach for (choosing) selecting the optimal ERT hyperparameters that can achieve the highest classification accuracy using particle swarm optimization (PSO). PSO [22] is a type of evolutionary algorithm that is commonly used to solve optimization problems. The PSO algorithm simulates the individual and social behaviors of a biological population [23]. PSO's central idea is to begin with a stochastic (random) point and then assess that particle's fitness in order to perform iterative optimization to find the optimal answer [24]. The PSO is a global incremental (iterative) optimization technique in which each solution (individual) in the population is represented as a particle. Initially, a fitness function is calculated to assess the fitness value of each (solution) particle, and these (solution) particles are supposed to be moving inside the solution space based on their position and speed, attempting to follow the present of ideal moving particles [25]. A final best solution will be found as a result of this process. Throughout each iteration, the (solution) particle will follow a path that is optimal for both itself or the group. Each particle defines a potential solution to the problem under consideration [26]. These particles move through the search space in search of the best optimal solution to a given problem. Each particle connects with other particles inside the search_space to get the best (optimal) solution. Position and velocity are two important factors that influence particle performance. Before delivering the final solution, multiple iterations are usually performed. The following equations are used to calculate the particle's velocity and position:

$$v_i^{t+1} = wv_i^t + c_1r_1(pbest_i^t - p_i^t) + c_2r_2(gbest^t - p_i^t) \quad (1)$$

$$p_i^{t+1} = p_i^t + v_i^{t+1} \quad (2)$$

where i is a single individual (particle) in the search space, c1 and c2 are acceleration coefficients, r1 and r2 are random numbers, w is the inertial coefficient, t is the iteration, v is the particle velocity at iteration t, p is the particle position in iteration t, p best is the I particle best position, and g best is the global best solution at time t, which will contain the best.

Each PSO iteration typically consists of three steps. The fitness function of each particle is computed. The local and global best for each particle of the PSO optimization are then updated. Finally, the velocity and position of each particle in the iteration are upgraded. This procedure is iterated until the termination condition is achieved. Algorithm 1 depicts the algorithm and the steps taken to select the hyperparameters. PSO is initially seeded with random particles (each particle consists of set of hyperparameters) containing hyperparameter details in the first step. In ERT there are many hyperparameters that need to be optimized (tuned) in order to have an efficient classifier. The Table 1 shows the most important hyperparameters which have been optimized. The accuracy of the model trained on particle values was represented by the fitness function. The accuracy of each (individual) particle is assessed. The particles then communicate with one another in order to find the best global solution.

Algorithm 1. Steps of hyperparameters optimization based on PSO
```
Input: Available ERT hyperparameters.
Output: The best possible combination of hyperparameters that achieve the best possible
accuracy.
1.  Create random particles (each particle consists of set of hyperparameters)
2.  evaluate the fitness (accuracy) of each particle
3.      Train ERT model
4.      Validate ERT model
5.  update local and global best
6.  update particles positions and velocities
7.  check termination condition
8.  if termination condition not met
9.      repeat step 2
10. else
11. return hyperparameters
12. end
```

Table 1. Hyperparameters (particles) and their configuration space

Hyperparameters	Type	Search space
n_estimators	discrete	[5,100]
max_depth	discrete	[4,40]
min_samples_split	discrete	[2,10]
min_samples_leaf	discrete	[2,10]
max_features	discrete	[1,22]

3.3. Classification using proposed an optimized extremely randomized trees model

Extremely randomized trees (ERT) [27], also defined as extra trees, is a tree-based ensemble method for supervised categorization (classification) and (prediction) regression. The Extra_Trees algorithm employs the traditional top-down procedure to construct an_ensemble of unpruned regression or decision trees. The main differences between extra_trees and other tree_based ensemble_methods are that it divides nodes by selecting cut_points completely at random and it uses the entire training_data to train each decision tree (rather than a bootstrap replica) to grow the trees while minimizing bias and variance. In the worst scenario, the method selects one characteristic (feature) and one cut/split point for each branch [28]. The Extra-Trees classifier seems to be less computationally expensive than other tree-based ensemble algorithms because it splits nodes by selecting cut-points completely at random. Extra-Trees are made up of a large number of separate decision trees that are grown using the entire training dataset. A decision_tree is consisting of a root_node, child_nodes, and leaf_nodes. The last decision is obtained based on the prediction of each tree by a majority of votes in classification problems [29]. ERT chooses a split rule at the root node based on a stochastic (random) subset of features and a pseudorandom cut point. This procedure is reiterated (repeated) until a leaf_node is reached. In ERT there are many hyperparameters that need to be optimized (tuned) in order to have an efficient classifier. After conducting many experiments, we found that the most important parameters that affect the efficiency of the model can be outlined as: the number of trees in the ensemble [n_estimators in sklearn python library], the number of features/attributes to select randomly [max_fetures in sklearn python library], and the minimum number of samples/instances required to split a node [min_sample_split in sklearn python library], longest route from the leaf node to the root node [max_depth in sklearn python library], after splitting a node, there should be a minimal amount of samples (instances) present in the leaf node [min_sample_leaf in sklearn python library]. So, we used the PSO algorithm to get the optimal values for hyperparameters that make the classification accuracy the best possible.

4. RESULTS AND DISCUSSION

This section summarizes the results of experiments and the outcomes of our proposed machine learning model, as well as comparing the proposed model's findings to other studies. The model's performance was evaluated by many metrics including (Accuracy, Specificity, Sensitivity, Precision and F1-score). Binary classification was done on the public "DDoS attack SDN dataset" obtained from the SDN environment contains 104 thousand records, and it was done using the sklearn python machine learning framework.

4.1. Performance metrics

On the dataset, the proposed machine learning model was trained and tested. We calculated Accuracy (ACC), Specificity (Sp), Sensitivity (Se), Precision (Pr) and F1-score for the model to assess its performance [30]. A confusion_matrix is shown in Table 2; the above-mentioned performance parameters are frequently calculated using a confusion matrix. In a binary classification problem, it is defined as a 2*2 matrix containing the classifier's actual and predicted values [31]. The four values in the matrix are somewhat perplexing, as explained in the Table 3.

Table 2. Confusion matrix

	Predicted positive	Predicted negative
Actual positive	TP	FP
Actual negative	TN	FN

Table 3. Elements of confusion matrix

Element	Defintion
True positive $_{tp}$ (normal)	A true positive value is one in which both the model prediction and the true values in the dataset are positive.
True negative $_{tn}$ (attack)	The true negative value is one in which both the model prediction and the actual values in the dataset are negative.
False positive $_{fp}$ (Misclassified as normal)	False positive errors occur when the model predicts a positive result while the actual values in the dataset show a negative result.
False negative $_{fn}$ (Misclassified as attack)	False negative errors occur when actual values in the dataset are positive but the model predicts something negative.

The above-mentioned performance parameters obtained from confusion matrix were used to evaluate the suggested model. These metrics' formulas are as:

$$ACC = \frac{TP+TN}{TP+TN+FP+FN} \quad (3)$$

$$Sp = \frac{TN}{TN+FP} \quad (4)$$

$$Se = \frac{TP}{TP+FN} \quad (5)$$

$$F1 - score = \frac{2\times se\times pr}{se+pr} \quad (6)$$

$$Pr = \frac{TP}{TP+FP} \quad (7)$$

4.2. Performance evaluation based on proposed an optimized extremely randomized trees

In this stage after the pre-processing, the dataset was partitioned (split) into two parts: testing and training at a rate of 0.3 and 0.7, respectively, at first, we applied and tested the standard ERT model with the default hyperparameters. Table 4 shows its results, and Figure 2 shows the normalized confusion matrix for standard ERT model with the default hyperparameters. After that, the PSO algorithm was used to optimize and select the hyperparameters of the ERT algorithm. After performing the hyperparameter optimization process using the PSO algorithm, the following hyperparameter values (n_estimators=59, max_depth=31, min_samples_split=4, min_samples_leaf=9, max_features=20) were obtained and used to build the proposed model used in the classification process. Then the model was used to classify the traffic and tested on the test_data, and a high accuracy rate of 99.95 was obtained, the classification results are shown in Table 5, and the normalized confusion matrix of the proposed model is shown in Figure 3.

Table 4. Classification results of the standard ERT model

Acc (%)	Sp (%)	Se (%)	F1-score (%)	Pr (%)
94.19	97.50	91.72	93.48	98.01

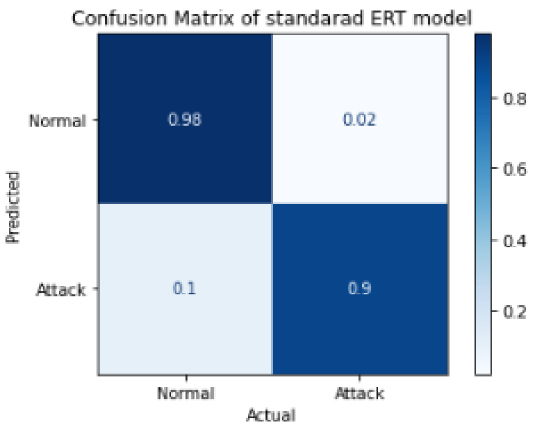

Figure 2. Normalized confusion matrix of the standard ERT model

Table 5. Classification results of an optimized ERT model

Acc (%)	Sp (%)	Se (%)	F1-score (%)	Pr (%)
99.95	99.90	100	99.95	99.91

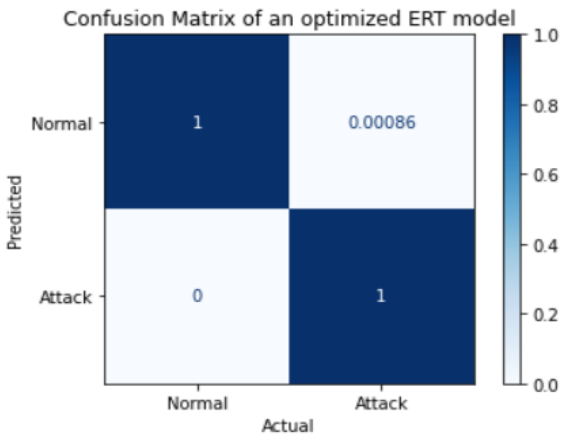

Figure 3. Normalized confusion matrix of an optimized ERT model

To evaluate the paper's proposed method, Table 6 compares the results of research on DDoS traffic detection using ML techniques with the model we propose. When looking at Table 6, it's clear that several datasets were utilized to detect attack traffic. Many of the authors employed public datasets to detect attack traffic in SDN but this datasets including network traffic statistics from classical network architectures, such as NSL-KDD, UNB-ISCX, CAIDA2016, CICIDS2017, and KDD Cup'99 [11], [12], [15]. Such datasets are helpful for evaluating the effectiveness of ML techniques used in determining malicious traffic. On the other hand, in addition to the current attack vectors, the SDN design has its own set of attack vectors because it is different from conventional network architecture. Furthermore, the growing volume and variety of attack traffic necessitates the usage of new datasets. Due to this, studies [16], [17], [30] utilization datasets collected in SDN architecture in their researches.

In Table 6, the top existing benchmark results is found to be 98.8% and 98.9%. As can be observed, the accuracy of our suggested model is the highest at 99.95%. The improved (optimized) approach followed for the suggested model is useful for detecting attacks. Our work aims to contribute to the research being conducted in this field (assaults detection in SDN utilizing machine-learning and optimization techniques).

The use of PSO algorithm for hyperparameters optimization improved the accuracy of machine learning approaches in identifying attack traffics, according to our results. Experimental studies were conducted by selecting the hyperparameters automatically by using the PSO algorithm to choose the appropriate values for the hyperparameters that make the model accuracy as best as possible. When combined with hyperparameters optimization algorithms, it can be argued that the model's classification performance positively affects the assault classification.

Table 6. A comparison of relevant work

Datasets and Related studies	ML techniques	Accuracy (%)
KDD Cup'99 [11]	DNN and SVM classifiers	92.30
CAIDA2016 [12]	Hybrid of KNN and SOM techniques	98.24
NSL-KDD [15]	Naive Bayes, C4.5, decision table, Bayesian network	91.68
Their Dataset [16]	An evolutionary SVM	98.90
Their Dataset [17]	Linear and polynomial SVM models	95
DDOS attack SDN Dataset [30]	LR, ANN, KNN, SVC, RF, Ensemble Classifier, hybrid SVC-RF	98.80
DDOS attack SDN Dataset	Our proposed an optimized ERT model	99.95

5. CONCLUSION

In this work, an optimized machine learning-algorithm was used to classify attack and normal traffic in a dataset generated from an SDN environment. The dataset contains 1,04,345 records and 23 features and consist of (UDP, TCP and ICMP) protocols as attack and normal traffics. The data includes numerical (statistical) features such as packet rate, byte count, packet per flow and duration-sec, in addition to features that indicate source and destination devices. The PSO algorithm was used to perform efficient classification and select the most appropriate hyperparameters for the ERT model. After conducting several experiments, the most hyperparameters that affect the efficiency of the machine learning model were extracted, the optimal values for these hyperparameters were defined using the PSO algorithm. After preprocessing and hyperparameter optimization, the suggested method was able to classify over 100,000 network records. According to the outcomes of the tests, the proposed model has a 99.95% accuracy rate. In the future, it is planned to use the deep learning models' and raise the diversity of assaults and discuss the classification efficiency and performance.

REFERENCES

[1] Y. Zhang, L. Cui, W. Wang, and Y. Zhang, "A survey on software defined networking with multiple controllers," *JJournal of Network and Computer Applications*, vol. 103, pp. 101–118, 2018, doi: 10.1016/j.jnca.2017.11.015.

[2] H. H. Khairi, S. H. S. Ariffin, N. M. A. Latiff, K. M. Yusof, M. K. Hassan, and M. Rava, "The impact of firewall on TCP and UDP throughput in an openflow software defined network," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 20, no. 1, pp. 256–263, 2020, doi: 10.11591/ijeecs.v20.i1.pp256-263.

[3] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes, "Software-defined networking security: Pros and cons," *IEEE Communications Magazine*, vol. 53, no. 6, pp. 73–79, 2015, doi: 10.1109/MCOM.2015.7120048.

[4] E. Amiri, E. Alizadeh, and M. H. Rezvani, "Controller selection in software defined networks using best-worst multi-criteria decision-making," *Bulletin of Electrical Engineering and Informatics (BEEI)*, vol. 9, no. 4, pp. 1506–1517, Aug. 2020, doi: 10.11591/eei.v9i4.2393.

[5] M. H. H. Khairi, S. H. S. Ariffin, N. M. A. Latiff, and K. M. Yusof, "Generation and collection of data for normal and conflicting flows in software defined network flow table," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 22, no. 1, pp. 307–314, Apr. 2021, doi: 10.11591/IJEECS.V22.I1.PP307-314.

[6] B. Mladenov and G. Iliev, "Optimal software-defined network topology for distributed denial of service attack mitigation," *Bulletin of Electrical Engineering and Informatics (BEEI)*, vol. 9, no. 6, pp. 2588–2594, Dec. 2020, doi: 10.11591/eei.v9i6.2581.

[7] D. Mukhopadhyay, B.-J. Oh, S.-H. Shim, and Y.-C. Kim, "A study on recent approaches in handling DDoS attacks," Dec. 2010, Accessed: Apr. 18, 2022. [Online]. Available: http://arxiv.org/abs/1012.2979.

[8] L. F. Eliyan and R. Di Pietro, "DoS and DDoS attacks in software defined networks: a survey of existing solutions and research challenges," *Future Generation Computer Systems*, vol. 122, pp. 149–171, 2021, doi: 10.1016/j.future.2021.03.011.

[9] N. Ahuja, G. Singal, and D. Mukhopadhyay, "DDOS attack SDN dataset," *IEEE Access*, vol. 1, 2020, doi: 10.17632/JXPFJC64KR.1.

[10] M. A. Naagas, E. L. Mique, T. D. Palaoag, and J. S. Dela Cruz, "Defense-through-deception network security model: Securing university campus network from DOS/DDOS attack," *Bulletin of Electrical Engineering and Informatics (BEEI)*, vol. 7, no. 4, pp. 593–600, Dec. 2018, doi: 10.11591/eei.v7i4.1349.

[11] B. V. Karan, D. G. Narayan, and P. S. Hiremath, "Detection of DDoS attacks in software defined networks," *2018 3rd International Conference on Computational Systems and Information Technology for Sustainable Solutions (CSITSS)*, Dec. 2018, pp. 265–270, doi: 10.1109/CSITSS.2018.8768551.

[12] T. M. Nam *et al.*, "Self-organizing map-based approaches in DDoS flooding detection using SDN," *2018 International Conference on Information Networking (ICOIN)*, Apr. 2018, pp. 249–254, doi: 10.1109/ICOIN.2018.8343119.

[13] K. Adhikary, S. Bhushan, S. Kumar, and K. Dutta, "Decision tree and neural network based hybrid algorithm for detecting and preventing ddos attacks in VANETS," *International Journal of Innovative Technology and Exploring Engineering.*, vol. 9, no. 5, pp. 669–675, Mar. 2020.

[14] R. M. A. Ujjan, Z. Pervez, K. Dahal, A. K. Bashir, R. Mumtaz, and J. González, "Towards sFlow and adaptive polling sampling for deep learning based DDoS detection in SDN," *Future generation computer systems*, vol. 111, pp. 763–779, Oct. 2020, doi: 10.1016/J.FUTURE.2019.10.015.

[15] S. Nanda, F. Zafari, C. Decusatis, E. Wedaa, and B. Yang, "Predicting network attack patterns in SDN using machine learning approach," *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, May 2017, pp. 167–172, doi: 10.1109/NFV-SDN.2016.7919493.

[16] K. S. Sahoo, B. K. Tripathy, K. Naik, S. Ramasubbareddy, B. Balusamy, M. Khari, and D. Burgos, "An evolutionary SVM model for DDOS attack detection in software defined networks," *IEEE Access*, vol. 8, pp. 132502–132513, 2020, doi: 10.1109/ACCESS.2020.3009733.

[17] A. T. Kyaw, M. Z. Oo, and C. S. Khin, "Machine-learning based DDOS attack classifier in software defined network," in *2020 17th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, 2020, pp. 431–434, doi: 10.1109/ECTI-CON49241.2020.9158230.

[18] Ö. Tonkal, H. Polat, E. Başaran, Z. Cömert, and R. Kocaoğlu, "Machine learning approach equipped with neighbourhood component analysis for ddos attack detection in software-defined networking," *Electronics*, vol. 10, no. 11, 2021, doi: 10.3390/electronics10111227.

[19] A. E. Cil, K. Yildiz, and A. Buldu, "Detection of DDoS attacks with feed forward based deep neural network model," *Expert Systems with Applications*, vol. 169, no. April 2020, p. 114520, 2021, doi: 10.1016/j.eswa.2020.114520.

[20] M. Myint Oo, S. Kamolphiwong, T. Kamolphiwong, and S. Vasupongayya, "Advanced support vector machine (ASVM) based detection for distributed denial of service (DDoS) attack on software defined networking (SDN)," *Journal of Computer Networks and Communications*, vol. 2019, 2019, doi: 10.1155/2019/8012568.

[21] J. Manan, A. Ahmed, I. Ullah, L. Merghem-Boulahia, and D. Gaïti, "Distributed intrusion detection scheme for next generation networks," *Journal of Network and Computer Applications*, vol. 147, p. 102422, Dec. 2019, doi: 10.1016/J.JNCA.2019.102422.

[22] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, 1998, vol. 1447, pp. 591–600, doi: 10.1007/BFB0040810.

[23] Q. Yao *et al.*, "Taking human out of learning applications: a survey on automated machine learning," *arXiv preprint arXiv:1810.13306*. Oct. 2018, doi: 10.48550/arxiv.1810.13306.

[24] M. H. Ahmad, K. Osman, and S. I. Samsudin, "Design of proportional integral and derivative controller using particle swarm optimization technique for gimbal system," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 26, no. 2, pp. 714–722, 2022, doi: 10.11591/ijeecs.v26.i2.pp714-722.

[25] M. I. Berbek and A. A. Oglah, "Adaptive neuro-fuzzy controller trained by genetic-particle swarm for active queue management in internet congestion," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 6, no. 1, pp. 229–242, 2022, doi: 10.11591/ijeecs.v26.i1.pp229-242.

[26] S. H. Shri and A. F. Mijbas, "Chaotic theory incorporated with PSO algorithm for solving optimal reactive power dispatch problem of power system," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 22, no. 3, pp. 1739–1747, 2021, doi: 10.11591/ijeecs.v22.i3.pp1739-1747.

[27] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Machine Learning*, vol. 63, no. 1, pp. 3–42, 2006, doi: 10.1007/s10994-006-6226-1.

[28] M. R. C. Acosta, S. Ahmed, C. E. Garcia, and I. Koo, "Extremely randomized trees-based scheme for stealthy cyber-attack detection in smart grid networks," *IEEE Access*, vol. 8, no. Ml, pp. 19921–19933, 2020, doi: 10.1109/ACCESS.2020.2968934.

[29] U. Saeed, S. U. Jan, Y. D. Lee, and I. Koo, "Fault diagnosis based on extremely randomized trees in wireless sensor networks," *Reliability Engineering & System Safety*, vol. 205, no. May 2020, p. 107284, 2021, doi: 10.1016/j.ress.2020.107284.

[30] N. Ahuja, G. Singal, D. Mukhopadhyay, and N. Kumar, "Automated DDOS attack detection in software defined networking," *Journal of Network and Computer Applications*, vol. 187, p. 103108, Aug. 2021, doi: 10.1016/J.JNCA.2021.103108.

[31] A. K. Hilool, S. H. Hashem, and S. H. Jafer, "Intrusion detection system based on bagging with support vector machine," *Indones. Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 24, no. 2, pp. 1100–1106, 2021, doi: 10.11591/ijeecs.v24.i2.pp1100-1106.

BIOGRAPHIES OF AUTHORS

Hasan Kamel received his Bachelor degree in Computer Engineering, from Al Mustansiriyah University, Iraq in 2018. His interests involve Computer Networks and Artificial Intelligence. He can be contacted at email: egma003@uomustansiriyah.edu.iq.

Mahmood Zaki Abdullah is one of an associate professor doctor at Mustansiriya University Baghdad-Iraq. He holds a PhD degree in computer engineering and has many books in computer engineering, networks, smart systems and information technology. He can be contacted at email: drmzaali@uomustansiriyah.edu.iq.