Journal of Network and Computer Applications 225 (2024) 103868

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

Research paper

Securing SDN: Hybrid autoencoder-random forest for intrusion detection and attack mitigation

Lotfi Mhamdi *, Mohd Mat Isa

School of Electronic and Electrical Engineering, University of Leeds, Leeds, LS2 9JT, West Yorkshire, United Kingdom

ARTICLE INFO

Keywords:
Network security
Software defined networking
Machine learning
Intrusion detection
Autoencoder and random forest

ABSTRACT

Software Defined Networking (SDN) has revolutionized network administration by providing centralized management through software, enabling traffic adjustment independent of the data plane. Despite the benefits, SDN networks are prone to security threats from external sources, thus necessitating the implementation of security measures. Unfortunately, most existing efforts have been just a simple mapping of earlier solutions into the SDN environments. This paper addresses the problem of SDN security based on deep learning in a purely native SDN environment, where a Deep Learning intrusion detection module is tailored to a native SDN environment. In particular, we propose a hybrid Deep AutoEncoder with a Random Forest classifier model (DAERF) to enhance intrusion detection performance in a native SDN environment. The proposed model is incorporated into a novel adaptive framework for attack mitigation in SDN environments. The proposed framework consists of a three-layer protection mechanism for detecting and preventing attacks. It is based on entropy-based detection, hybrid machine learning in the control layer and proactive services monitoring in the application layer. Experimental results have shown that our DEARF proposed autoencoder model achieved anomaly detection rates in excess of 98% in stand-alone mode as well as when incorporated within the framework, making it highly solution for next generation SDN networks.

1. Introduction

The Software Defined Network (SDN) paradigm is a modern approach to network structures that distinguishes itself from traditional networking. In an SDN architecture, software-based controllers and Application Programming Interfaces (API) are utilized to interact with switches and routers, as well as to control access (Nunez et al., 2023). This approach adopts a three-plane architecture which includes the management plane, control plane, and data plane. The data plane is made up of network switches while the control plane comprises a controller core that communicates with the switches. The APIs around the controller core serve as a means to interact with the different layers (Nunez et al., 2023; Kreutz et al., 2014). The SDN architecture allows for the centralized management and control of the network, making it easier to manage and optimize network performance. SDN also enables network administrators to define network policies and rules, create virtual networks, and provide flexibility and customization options using APIs. With its advanced architecture, SDN provides a sophisticated and modern approach to network structures.

The SDN architecture has several advantages over traditional networking. However, its susceptibility to attacks and the potential failure of the entire network if the controller is compromised pose serious security threats (Nunez et al., 2023). To overcome these security issues, it is crucial to secure networks and keep them safe for use. Intrusion detection is one of the most effective ways to combat security threats (Uppal et al., 2014).

Intrusion detection techniques can be classified into two categories: signature-based and behavior-based. Signature-based detection uses predefined patterns of attack signatures, while behavior-based detection relies on the analysis of network traffic behavior. In SDN, different types of intrusion, such as unauthorized access, data leakage, data modification, compromised applications, DoS, configuration issues, and system-level SDN security, can lead to network attacks (Zhao et al., 2019).

Machine Learning (ML) approaches provide an effective solution for detecting anomalous behavior that violates pre-defined rules such as unauthorized access or malware detection in SDN networks (Das et al., 2015). By leveraging machine learning and deep learning techniques, network security can be improved, enabling a modern and advanced approach to network security (Sabeel et al., 2019). Deep learning techniques, such as Convolutional Neural Networks (CNNs) and Recurrent

* Corresponding author.
E-mail addresses: L.Mhamdi@leeds.ac.uk (L. Mhamdi), elmsbm@leeds.ac.uk (M.M. Isa).
URL: https://eps.leeds.ac.uk/electronic-engineering/staff/551/lotfi-mhamdi (L. Mhamdi).

https://doi.org/10.1016/j.jnca.2024.103868
Received 12 November 2023; Received in revised form 6 February 2024; Accepted 12 March 2024
Available online 15 March 2024

Neural Networks (RNNs), have shown promising results in detecting various network attacks, including DoS and DDoS (Zhang et al., 2019a; Tang et al., 2018; Liu et al., 2022).

Extensive research works have focused on early detection, such as counting the number of connections, the entropy of transactions, and others (Halder et al., 2017; Lim et al., 2014; Mousavi and St-Hilaire, 2015). Embarking on machine learning capabilities is also being explored with the aim of studying the data representation and explicit meanings (Tang et al., 2018). Prevention steps and action are also being experimented with, aiming to minimize the impact of, and if possible, repel any attacks (Sattar et al., 2016; Kumar et al., 2018; Huang et al., 2020).

Another effective approach for intrusion detection is to combine the Autoencoder (AE) and Random Forest (RF) algorithms. The AE algorithm can identify the characteristics of normal network traffic behavior. The RF algorithm can then classify the traffic based on the features identified by the AE and determine whether it is normal or malicious. This technique has demonstrated high accuracy in detecting network intrusion. The AE and RF algorithm combination is another effective approach for detecting network intrusion, particularly in identifying new or unknown attacks.

With the focus on integrating and combining the different proposals into a complete approach, this paper proposes a unified adaptive framework for attack mitigation in a native SDN environments. In particular, the key difference in our proposed work is the build up of several detection and prevention blocks into one consolidated framework which covers protection in different layers of the SDN environment. In summary, the contributions of this paper are:

- The introduction of a hybrid combination of a deep autoencoder (DAE) and random forest (RF) algorithm in the native SDN environment, deployed in the SDN controller mechanism. Within the SDN controller, our proposed DAERF approach yields a detection rate in excess of 98% using purely native SDN statistics collection as features.
- The description of a novel three-layer adaptive framework for attack mitigation in native SDN environments. It consists of entropy-based detection, hybrid machine learning in the control layer and proactive services monitoring in the application layer. Experimental results show that DAERF achieves more than 98% detection rate in the proposed IDS framework.

The rest of this paper is structured in the following pattern. In Section 2, a thorough review of the relevant literature and previous research in the field will be provided to establish the context for the current study. The methodology used in the study, including the research design, data collection methods, and statistical analysis techniques, will be discussed in Section 3. Section 4 will present the study's results and performance analysis. These findings will be compared to previous research in the field, offering a critical evaluation of the study's contributions. Finally, Section 5, concludes the paper by summarizing the main findings of the study, discussing their implications, and suggesting directions for future research.

2. Related work

In recent years, intrusion detection has become increasingly important due to the rise of cyber attacks. Researchers have proposed various methods for detecting intrusions, including the use of deep learning, big data visualization and statistical analysis, and entropy-based solutions.

One such method is the non-symmetric deep autoencoder (NDAE) model proposed by Shone et al. (2018), which achieved an 85.42% accuracy rate in detecting intrusions. Another approach utilized big data visualization and statistical analysis in combination with an autoencoder, resulting in an accuracy rate of 87% (Ieracitano et al., 2018). In Zhang et al. (2019a), a neural network-based anomaly detection system was introduced, combining Lenet5 convolutional neural network and Long Short-Term Memory (LSTM) network, achieving an impressive accuracy rate of 99%. In Yang et al. (2022), a real-time network intrusion detection system via ensemble of Autoencoder in SDN was proposed with an accuracy of 97%.

Deep Neural Network (DNN) was used in Tang et al. (2018) with three hidden layers, which performed binary classifications based on the NSL-KDD dataset with six basic feature selections, achieving an accuracy rate of 75.75%. In Ferrag and Maglaras (2019), a novel deep learning and energy platform named DeepCoin was designed using blockchain technology, with an accuracy level of 98.23%. Another intrusion detection system (IDS) was proposed in Binbusayyis and Vaiyapuri (2019), using the Random Forest learning algorithm, with an impressive accuracy rate of up to 99.88%.

In the context of SDN, Ye et al. (2018) proposed a method for detecting attacks using two characteristic values, namely the speed of source IP (SSIP) and the ratio of pair flow (RPF), combined with Support Vector Machine (SVM). They extracted a six-characteristic value matrix and processed it from multidimensional and low-dimensional nonlinear into high-dimensional feature space, enabling the SVM model to achieve an average accuracy of 95.24%.

Another approach for detecting Distributed Denial of Service (DDoS) attacks in SDN environments was proposed in Fan et al. (2021a,b). Both studies utilized entropy-based solutions to measure attack behavior occurrences in the network. Fan et al. (2021a) proposed a fusion entropy technique to differentiate between legitimate and anomalous traffic, and combined information entropy and log energy entropy to detect DoS attacks. Their experimental findings showed that the entropy value for attack scenarios was 91.25%. Similarly, in Fan et al. (2021b), the entropy property was used to distinguish between legitimate and anomalous traffic, with a combination of information entropy and log energy entropy used to detect DDoS attacks. The experimental results showed that the entropy value of the attack scenarios was 91.25% lower than normal scenarios. These entropy-based solutions provide several advantages and are more significant compared to other attack detection methods.

A study proposed by Ashraf and Latif (2014) aimed to improve the processing delay of legitimate nodes in the network by calculating random flow data using entropy. The researchers utilized Mininet, an open-source network emulator, and the extension module within the Floodlight SDN controller to simulate the process. The data was collected based on the destination IP information and a few attributes of the TCP flags. The results of the study showed an impressive improvement of 13% in the processing delay of legitimate nodes. The performance of CPU use and attack detection time were also analyzed, indicating a positive impact in the tested scenarios. The use of entropy in calculating random flow data showed promising results in improving the efficiency of the network.

In another approach proposed by Ashraf and Latif (2014), the researchers employed Artificial Neural Network-based (ANN) methods to detect Distributed Denial of Service (DDoS) attacks. A dynamic Multilayer Perceptron (MLP) with a feedback mechanism was utilized to identify both known and unknown DDoS assaults. The MLP was trained on a range of attributes that could distinguish between normal and attack traffic flows. This approach proved to be effective in detecting DDoS attacks with high accuracy.

A trigger system to detect DDoS attacks more efficiently and reduce stress on switches was proposed in Braga et al. (2010). The researchers utilized the controller's control plane trigger mechanism to detect attacks, which successfully reduced the strain on switches. However, the authors noted that this trigger system increased the strain on the controller to some extent. Despite this, the proposed system was effective in detecting DDoS attacks in a timely manner while minimizing the impact on the network's resources. This trigger system could be further optimized to reduce the strain on the controller, making it an efficient solution for detecting DDoS attacks.

In the context of SDN, DDoS attacks can cause significant harm to the network infrastructure. Therefore, the authors proposed a trigger system in their research work (Wang et al., 2019), which is capable of quickly identifying DDoS attacks while reducing the stress on the switches. The controller's control plane trigger mechanism is implemented to achieve this goal. While this approach has been found to be successful in mitigating DDoS attacks, it has been observed that it can increase the strain on the controller, which needs to be taken into account during its implementation. The trigger system allows for the rapid detection and response to DDoS attacks, reducing the chances of the attack causing significant damage to the network infrastructure. The system's design helps in maintaining the overall network performance, and by reducing the stress on the switches, it ensures that they are not overwhelmed by the DDoS traffic.

In their study, Abou El Houda et al. (2021) presented a novel framework for addressing network security risks in SDN. The proposed framework consists of two modules - a data flow collection module that employs the sFlow protocol and an unsupervised machine learning (ML) module for outlier detection. This approach significantly reduces the computational complexity while outperforming existing state-of-the-art methods in terms of accuracy and detection rate. The framework offers great potential in addressing emerging network security threats in SDN environments. With the ability to accurately detect and identify outlier events, the framework can help in mitigating potential security breaches and attacks. Furthermore, the unsupervised nature of the ML module allows for the detection of unknown and previously unseen threats, making it a valuable addition to the current arsenal of SDN security solutions.

In conclusion, as cyber attacks continue to rise, it is essential to have effective intrusion detection methods to safeguard the network infrastructure. As we shall see, the proposed unified 3-layer intrusion detection system coupled with autoencode and random forest algorithm (DAERF) have proven to be successful in detecting intrusions with impressive accuracy rates.

3. Methodology

This section introduces our framework for intrusion detection in SDN. We first describe the general overview of the system, then we give a detailed description of the proposed three-layer protection framework.

3.1. Basic architecture for intrusion detection firmware

The SDN intrusion detection is a critical component in maintaining the security of the entire network. The architectural overview of the native SDN intrusion detection system, as illustrated in Fig. 1, consists of two primary modules: the Assembly module and the Adaptive ML module (described in Section 3.2). The Assembly module is responsible for collecting statistical data from the network, which is used to identify patterns and anomalies in network traffic. It uses tools such as packet sniffers and flow analyzers to collect information about the network traffic. The collected data is then fed to the adaptive ML module, which is responsible for implementing deep learning algorithms to detect intrusion attempts based on unusual network activity. The adaptive ML module is capable of adapting to changing network conditions and identifying new and previously unseen threats. Once an intrusion is detected, the module enforces mitigation measures to prevent further damage and maintain the integrity of the network.

The Assembly module is a crucial element of network architecture that is responsible for collecting data from network traffic and forwarding it to the controller for analysis and decision-making. To achieve this, the Assembly module utilizes the OpenFlow standard protocol (McKeown et al., 2008) to send a stats request to all the switches in the network, which contains information about the state of the switches such as traffic flows, port statistics, and packet counts.

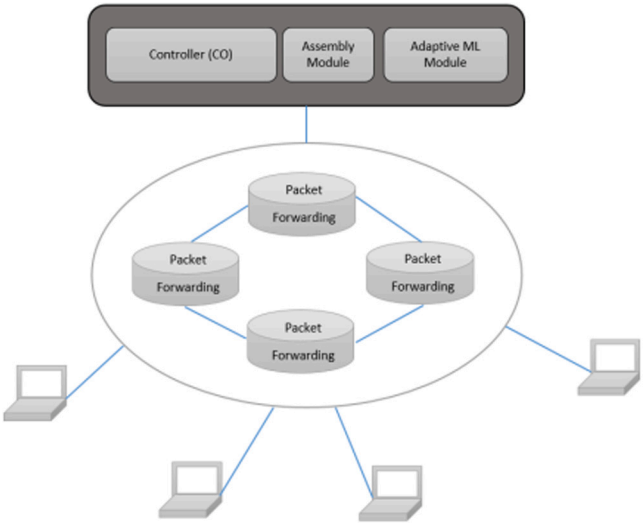

Fig. 1. Native intrusion mitigation architecture.

Once the stats request is sent, the switches respond with their current state information, which is then processed by the controller. This approach provides real-time visibility into network activity and allows the controller to identify potential anomalies or security threats and take appropriate action to mitigate risks.

The utilization of the OpenFlow standard protocol for data collection and analysis provides several advantages, such as a centralized view of network activity, which is essential for effective network management and security. Additionally, it allows the use of advanced analytics and machine learning techniques to identify patterns and anomalies in network traffic, resulting in improved accuracy and efficiency of intrusion detection and response.

3.2. Unified three-layer intrusion detection system

The proposed framework is shown in Fig. 2. The consolidated solution for the whole framework covers the layers within an SDN environment. The first layer, labeled number ①, is an entropy-based module that monitors traffic against attack attempts. The detailed working of this layer is described in Section 3.2.1. Traffic that has passed through the first layer is examined by the second module, which has hybrid machine learning intrusion detection located in the controller, as labeled in ②. The SDN controller keeps track of the services status of the public server being visited by the traffic in order to lessen the effects of an attack. Upon detecting a heavy load on the services, the controller checks both earlier layer detection to handle the load coming in. This layer is the core layer and is further described in Section 3.2.2. This part of the observation is deployed in the application layer, as shown in ③ of the proposed framework. This is further described in Section 3.2.3

3.2.1. Entropy based detection

One of the most significant security issues with SDN is the detection of low-rate DDoS attacks on the SDN controller. The difficulty of identifying the assault comes from the attack traffic's similarity to typical traffic behavior. It becomes much more difficult to achieve high accuracy levels and a low false-positive rate when many hosts are involved. Meanwhile, any detection technique must contend with high-rate DDoS attacks, especially when several targets are involved.

As a result, the suggested technique monitor the traffic packets in the SDN network passively to identify DDoS assaults on the SDN controller, independent of attack transactions and the amount of targets. A general Rényi joint entropy (Aladaileh et al., 2022) is adopted in this

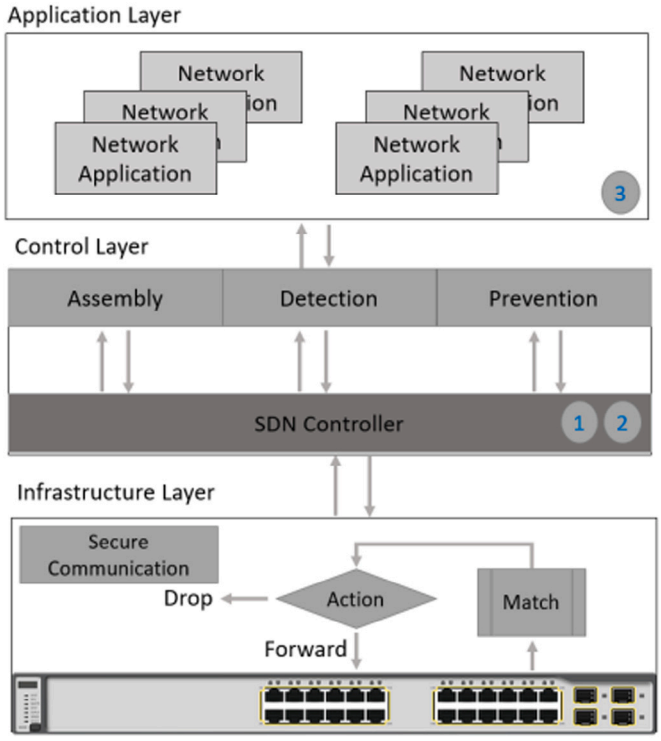

Fig. 2. Three-layer traffic monitoring.

study, by integrating two approaches: the joint entropy and the Rényi approach. The destination and source IP addresses, represented by y and x are measured in the general Rényi joint entropy in the form of two packet header characteristics. The Rényi joint entropy approach is a combinations of joint entropy and Rényi entropy, and can be derived as described below:

$$H(X) = -\sum_{x\epsilon X} p(x) log p(x) \tag{1}$$

$$H(X,Y) = -\sum_{x\epsilon X} -\sum_{y\epsilon Y} p(x,y) log p(x,y) \tag{2}$$

$$H_{RJ\alpha}(x) = \frac{1}{1-\alpha} log_2 \left(\sum_{i=1}^{N} \sum_{j=1}^{M} log p(x_i, y_j) \right) \tag{3}$$

where $H_{RJ\alpha}(x)$ is a Rényi joint entropy and $p(x_i, y_j)$ refers to the distribution of probability between the source IP x and destination IP y during the time interval of (t).

The Rényi joint entropy technique is dependent on a calculated value that could increase the detection rate by assessing the likelihood of traffic packets arriving. Based on the IP frequencies, the $p(x_i, y_j)$ distribution probability is derived for each source and destination IP address. When each packet's probability distribution is evenly spread throughout all the hosts' destinations, the Rényi joint entropy reaches its maximum value. The probability that all packets during a certain time window skewed towards a specific destination host produce a minimal value of Rényi joint entropy. The Rényi joint entropy is calculated using the likelihood that each source IP address (x_i) and destination IP address (y_j) were recorded in the previous step within a certain time frame. The outcome of this phase is to flag packets that are likely to cause DDoS attack threats.

3.2.2. The DAERF hybrid detection module

When building a typical machine learning models, essential input features are selected, and the model automatically learns by mapping

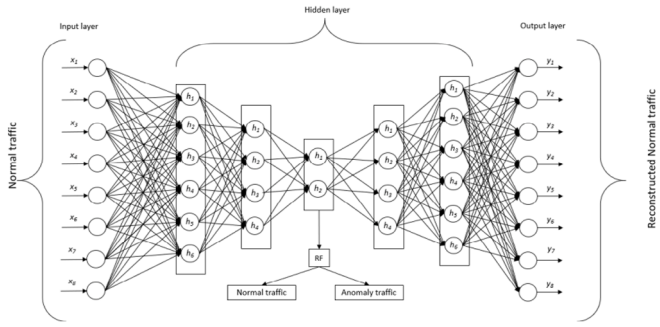

Fig. 3. DAERF: The Deep AutoEncoder with Random Forest model.

identified characteristics of the features to a conclusion output. In Deep AutoEncoder, there are multiple levels of encoding and decoding used. The abstract features from various levels are automatically being discovered and composed to produce output. The features from the previous level are carried forward to the next level to be processed again for another level of abstract representation. Our proposed DAERF model, as depicted in Fig. 3 consists of an input layer, three hidden layers, and an output layer. A total of 8 dimensions for input and output are selected in the autoencoder. The hidden layers contain six, four, and two neurons, respectively. The middle, hidden layer of two neurons is used to input the random forest classifier for the intrusion detection process, as shown in Fig. 3. A total batch size of 1000 were trained for ten epochs using the chosen Adam optimizer (Kingma and Ba, 2014).

Two phases are involved in the autoencoder, namely encoding and decoding. Input data that is named as x is compressed into h which is the low dimensional representation of the data x during the encoding process and then it is reconstructed into the input data during the decoding. In the Eqs. (4) and (5), the encoding is done in Eq. (4) where function $f(.)$ is a non-linearity activation function, while W and W' are weight matrices. The variable y is denoted as output and b and b' as biases.

$$h = f(Wx + b) \tag{4}$$

$$y = f(W'h + b') \tag{5}$$

During the training process, a loss function is used to measure the discrepancy (error) between the input data and the reconstructed output. This study uses the Mean Squared Error (MSE) as the loss function, as shown in Eq. (6), to train the autoencoder to identify important features and patterns in the input data.

$$L(x,y) = \|x - y\|_2^2 \tag{6}$$

To prevent the autoencoder from just copying the input data to the output, regularization constraints are enforced during the optimization process. Furthermore, to prevent overfitting of the data, an 8/2 compression ratio is utilized, which compresses the input data into a lower-dimensional representation and then reconstructs it to the original input data. The Rectified Linear Unit (ReLU) activation function is employed for both the hidden and output layers, which has demonstrated success in numerous deep learning applications. The output will be x if the output is not an intrusion and otherwise the output will be 0. This could be written as Eq. (7) below.

$$A(x) = max(0, x) \tag{7}$$

For optimization steps, regularization constraints are enforced to avoid the autoencoder overfitting the data by copying the input directly to the output. The architecture of the model is shown in Fig. 3, and Table 1 presents the model's parameters.

To perform classification, the output of the hidden layer of the autoencoder is processed and utilized as input for a classification

Table 1
Model parameters.

Input layer	Output layer	Hidden layers	Dropout	Act. Regulizer	Act. Funct.
8	8	6-4-2	0.5	10^{-5}	Relu

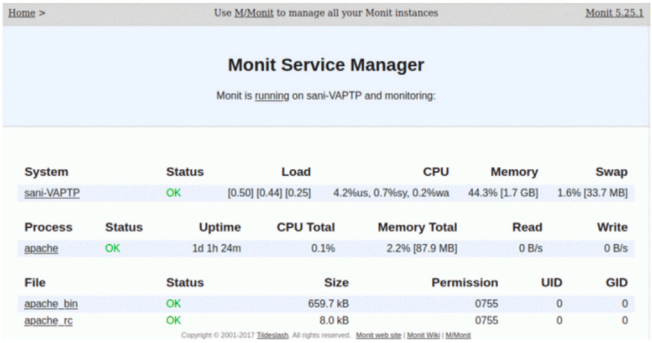

Fig. 4. Application layer traffic monitoring.

algorithm. In this study, the random forest algorithm is used because of its ability to handle high-dimensional data and control for overfitting. Random forest is an ensemble learning approach that involves subsampling the dataset and using averaging to improve the accuracy of the prediction model. The encoded output of the autoencoder is then fed into the Random Forest algorithm, which makes the final decision on whether the traffic is normal or an attack attempt.

The proposed hybrid algorithm has several advantages over traditional machine learning algorithms. The use of deep learning techniques allows the algorithm to learn complex features and patterns in the data, improving the system's ability to detect and classify different types of attacks accurately. The inclusion of the random forest classifier also enhances the algorithm's performance, as it enables the system to leverage the strengths of both deep learning and traditional machine learning techniques.

3.2.3. Passive application layer monitoring

In the third and final layer, a Ryu framework was adopted in our SDN topology. The controller plays two roles here: one as a standard SDN controller that controls and monitors the network, and the other as a defense mechanism. For the defense function, the controller keeps checking the status of the services being provided by the internal servers. A congested response from the servers via the status response indicates a high traffic flow and processing being handled by the targeted servers. In order to increase the response time for the server's services, heavily connected connections will be penalized to provide the affected server with ample processing time to recover from the impact of an attack. As shown in Fig. 4, we utilized Monit application (Tildeslash, 2000) for monitoring the status of the server for detection of poor response of provided services.

3.3. Dataset

Machine learning methods require a dataset to be well built up. We have used the CICIDS2017 dataset (Panigrahi and Borah, 2018) in our study, for various reasons. This dataset is designed explicitly for intrusion detection in SDN environments, and its comprehensive and up-to-date nature makes it an ideal choice for conducting research in this field. Furthermore, the dataset is regularly updated to include the latest types of attacks and network traffic. CICIDS2017 also contains a vast amount of labeled data, which is essential for the development and rigorous training of intrusion detection systems.

As stated in Sattar et al. (2016), the CICIDS2017 dataset provides the total payload packets, replicating actual network traffic activities in the targeted environment. This dataset covers seven types of common attack groups (i.e., Botnet, Brute Force Attack, DoS Attack, DDoS Attack, Heartbleed, Infiltration Attack, and Web Attack). This dataset consists of a Microsoft Excel CSV dataset and complete payload packets in a PCAP format file for a five day working hours period. The PCAP files are each sized 7 to 12 GB, respectively. Each flow sample of the dataset contains 83 flow features, explicitly explained in Chang et al. (2017). For this, a total of 16 native SDN OpenFlow flow features and a binary classification type were collected during the PCAP traffic emulation in a simulated SDN environment. The list of features is shown in Table 2.

Table 2
Native SDN collected features.

No.	Feature name	No.	Feature name
1	time	9	hard timeout
2	cookie	10	packet count
3	priority	11	byte count
4	reason	12	ip proto
5	table id	13	ipv4 src
6	duration sec	14	ipv4 dst
7	duration nsec	15	src port
8	idle timeout	16	dst port

Table 3
Selected SDN features.

No.	Feature name	No.	Feature name
1	duration sec	5	packet count
2	duration nsec	6	byte count
3	idle timeout	7	ip proto
4	hard timeout	8	dst port

Table 4
SDN environment flows collection.

Dataset	Attack	Normal	Total
Monday	0	263,156	263,156
Wednesday	16,622	210,469	227,271
Friday	1028	229,262	230,290
Total	17,650	702,887	720,717

Out of the 16 collected features, eight flow features have been selected for further machine learning analysis for intrusion detection in a native SDN environment. Another eight features have been dropped because the information differs for each network connectivity and does not represent the analyzed traffic pattern. The eight chosen features are listed in Table 3.

After the PCAP file had been injected into the SDN environment, there were a total of 263 156 collected flows for Monday, 227 271 collected flows for Wednesday, and 230 290 collected flows for Friday. The distribution of all three datasets according to network traffic classes is shown in Table 4.

Training and testing were prepared during the data processing phase. The collected dataset has an extensive range between the largest and the smallest values. For example, the biggest value for parameter duration is 999.10^6, while the smallest value is 0. A similar large range value also occurs in parameters such as packet count and byte count, which contribute to in-comparability and thus unsuitable processing. These values are normalized by using max–min normalization to map all the values in the range −1 to 1, according to:

$$x_i = \frac{x_i - Min}{Max - Min} \tag{8}$$

The values that were resulted from the normalization above were used in the model during the training and testing process. 20% of the total dataset was used for testing and the earlier selected 80%, for training.

4. Performance analysis

Before presenting the experiments, we would like to describe the structure of this Section. Before going into the experimental results, Section 4.1 first explains the evaluation metrics used throughout this work. The experimental results are divided into parts. The first part, Section 4.2, studies the performance of the DAERF algorithm presented in Section 3.2.2 in a stand alone mode. Then, Section 4.3 presents the results of our proposed unified IDS framework as presented in Section 3.2, referred to in the results as DAERF_IDS.

4.1. Evaluation metrics

The performance evaluation used in this study are the followings:

- **True Positive (TP)**: number of anomaly records correctly classified.
- **True Negative (TN)**: number of normal records correctly classified.
- **False Positive (FP)**: number of normal records incorrectly classified.
- **False Negative (FN)**: number of anomaly record incorrectly classified

These metrics include Recall (R), Precision (P), Accuracy (ACC), and F-measure (F). These measures are calculated by using the following formulas:

Recall (R) is the proportion of relevant instances that have been retrieved over the total number of relevant instances. The formula for Recall is:

$$R = \frac{TP}{TP + FN} \tag{9}$$

Precision (P) is the proportion of relevant instances that have been retrieved over the total number of instances that have been retrieved. The formula for Precision is:

$$P = \frac{TP}{TP + FP} \tag{10}$$

Accuracy (ACC) is the proportion of correct predictions made by the model over the total number of predictions made. The formula for Accuracy is:

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

F-measure (F) is the harmonic mean of Precision and Recall. It combines the two measures to provide a more balanced evaluation of the model's performance. The formula for F-measure is:

$$P = 2 \times \frac{P \times R}{P + R} \tag{12}$$

It is important to note that these evaluation metrics are commonly used in machine learning and natural language processing tasks to assess the effectiveness of different models. They provide a quantitative measure of the model's ability to correctly classify instances, and can be used to compare the performance of different models on the same task. Additionally, these metrics are also useful for identifying areas where the model may need further improvement, such as in cases where the Precision or Recall is low.

4.2. DAERF stand alone performance

Fig. 5 presents a visual representation of the performance of the DAERF model throughout the training and testing stages. A total of 80% of the Dataset (576,573 samples) were used for training and 20% (144,144 samples) for testing. Mininet (Kaur et al., 2014) was used as a network emulator with the Ryu (Li et al., 2020) component-based SDN as the framework. The proposed model simulation used TensorFlow (Bernico, 2018) back end. As can be seen from the Figure, the model exhibits a quite very good learning rate as can be seen in the decreasing loss curve over epochs.

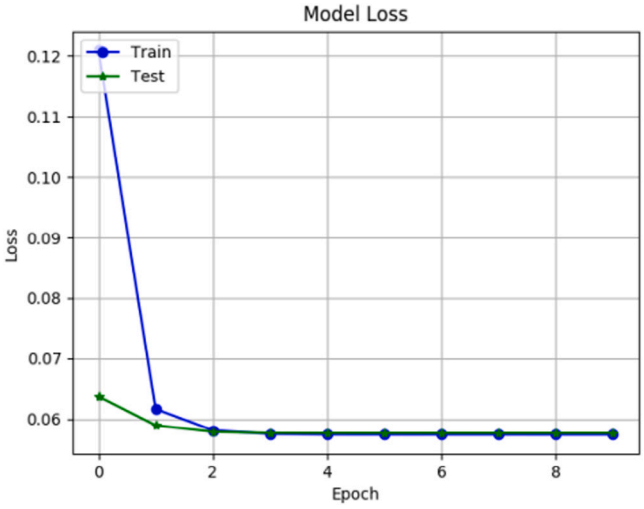

Fig. 5. DAERF testing and training performance.

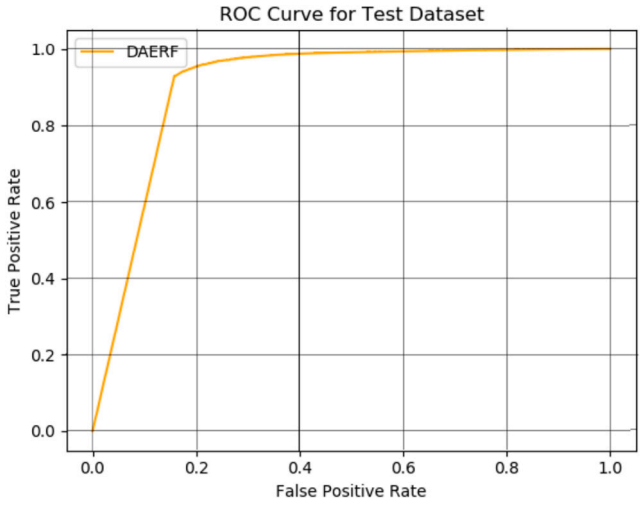

Fig. 6. DAERF ROC AUC curve.

Table 5
DAERF performance comparison.

Model	Accuracy	Precision	Recall	F1
Zhang et al. (2019b)	99.9	99.8	99.9	99.8
Binbusayyis and Vaiyapuri (2019)	99.8	99.9	99.8	–
Sharafaldin et al. (2018) Adaboost	–	77	84	77
Sharafaldin et al. (2018) ID3	–	98	98	98
Sharafaldin et al. (2018) KNN	–	96	96	96
Sharafaldin et al. (2018) MLP	–	77	83	76
Sharafaldin et al. (2018) Naive-Bayes	–	88	0.4	0.4
Sharafaldin et al. (2018) QDA	–	97	88	92
Sharafaldin et al. (2018) RF	–	98	97	97
DAERF	98	98.89	98.96	98.92

A standard measure for classifier comparison is the Receiver Operating Characteristic (ROC) curve. Plotting the false positive rate versus the true positive rate produces the curve for the ROC. The area under the curve (AUC) is a measure of determining a model's performance in predicting the classes. The higher the AUC, the better the classifier. This is confirmed in Fig. 6, the DAERF model achieves 0.92 (92%) AUC.

The performance of DAERF compared with other related research is shown in Table 5. The performance achieved was better than the

Table 6
Distribution of KDDTRAIN+ and KDDTEST+ Traffic classes.

Dataset	Normal	Dos	Probe	R2L	U2R	SUM
Train+	67,343	45,927	11,656	995	52	125,973
Test+	9711	7458	2754	2421	200	22,544

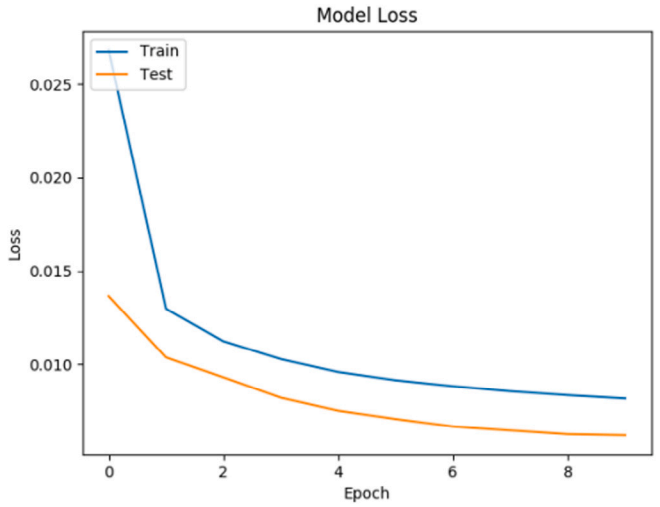

Fig. 7. DAERF testing and training performance using NSL-KDD dataset.

actual result from Sharafaldin et al. (2018). Note that Sharafaldin et al. (2018) did not report Accuracy measures, that is why we left it as '–' in the Table. DAERF has a slightly lower performance than Zhang et al. (2019b) and Binbusayyis and Vaiyapuri (2019). While this is minor, this slight lower performance is attributed to the difference in the scope of the experiment, whereby Zhang et al. (2019b) uses tshark for dumping and analyzing network traffic and Binbusayyis and Vaiyapuri (2019) uses the CICIDS2017 dataset in Weka. Having said that, the proposed DAERF still exhibits comparable performance.

4.2.1. DAERF with other datasets

In order to further test the DAERF performance, we have used the NSL-KDD dataset (Hong et al., 2021). The model is trianed on the KDDTrain+ dataset and evaluated on the KDDTest+ dataset. The KDDTest+ dataset is used to evaluate the performance of the model on days when no attacks occurred in the SDN environment, making it a reliable indicator of the model's ability to accurately detect attacks. The NSL-KDD dataset is a refined version of KDD-99 that addresses some of the issues with the original dataset, such as redundant and irrelevant features, and provides a more balanced distribution of data between normal and attack traffic. The statistics of the NSL-KDD dataset are shown in Table 6, which provides information about the number of instances in each class, the distribution of attacks, and other relevant details. This approach allows for a more comprehensive evaluation of the intrusion detection model's performance and its ability to accurately distinguish between normal and malicious traffic in an SDN environment.

The results of the model under the above settings have resulted into as high as 98.4% accuracy. This is even higher than that of CICIDS2017, as shown in Table 5. Fig. 7 depicts DAERF's performance over the testing and training stages, again with very good performance.

4.3. DAERF framework

In order to test DAERF within the three-layer framework, few steps are needed to have adequate features and measures. For this, we have again used the CICIDS2017 dataset (Panigrahi and Borah, 2018). This dataset includes seven common assault types (i.e. Botnet, Brute Force Attack, DoS Attack, DDoS Attack, Heartbleed, Infiltration Attack, and Web Attack). The dataset includes entire payload packets for a five-day work period as shown in Table 4. In our study, a combination of Wednesday and Friday afternoon with a DDoS attack dataset were chosen. Both days' dataset activities, which contain benign and also DoS/DDoS attacks were recorded. The selected features with sample size and class composition, are as shown in Table 7.

Table 7
Selected features in DAERF framework.

Traffic class	Label	Sample	Composition
BENIGN	BENIGN	537,749	58.550%
DDoS	DDoS	128,027	13.940%
DoS	DoS GoldenEye	10,293	1.120%
DoS	DoS Hulk	231,073	25.160%
DoS	DoS Slowhttptest	5499	0.598%
DoS	DoS Slowloris	5796	0.631%
Heartbleed	Heartbleed	11	0.001%
TOTAL	N/A	918,448	100%

Table 8
Comparison table for the native SDN intrusion detection framework.

Method	Accuracy rate	FPR
Ujjan et al. (2021) - CNN model	93%	9%
Ujjan et al. (2021) - SAE model	94%	6%
Carvalho et al. (2019)	95%	5%
DAERF_IDS	**98.16%**	**1.85%**

We ran the CICIDS2017 Wednesday and Friday dataset, which contains benign and DoS attacks, into the simulated DAERF_IDS framework. The controller's ability to detect DoS and DDoS attacks were tested during the entropy implementation. A window of 10-seconds was used to make the entropy calculation in order to determine whether traffic was benign or attacks. A total of 10 simulation runs were completed and the average rate for all 10 runs was reported. The average false positive rate was 1.85% while the average detection rate was 98.16%, as depicted in Table 8.

Table 8 demonstrates a comparison of the DAERF_IDS (framework) and other recent machine learning advancements for DDoS attack detection in SDN. The average detection rate and average false positive rate were used for comparison. As can be seen from the table, the DAERF_IDS framework produces better results as compared to earlier studies, for DDoS attack detection in SDN.

These results indicate that the proposed DAERF framework is very attractive in detecting and responding to malicious network traffic with a high level of accuracy and efficiency. The significance of selecting an appropriate dataset and optimizing the compression ratio to enhance the accuracy of the autoencoder and random forest-based intrusion detection systems in SDN is important in this regard.

4.4. DAERF SDN controller performance

In this Section, we evaluate the effect (overhead) of the DAERF on the SDN network controller performance. Below, we first describe the evaluation testbed and then the network performance evaluation is presented in terms of system's throughput and latency.

4.4.1. Experimental setup

The DAERF is implemented as an application within a Ryu controller. Cbench (Jawaharan et al., 2018) is a standard tool used for evaluating the SDN controller performance which supports two running modes, throughput and latency. The throughput mode computes the maximum number of packets handled by the controller and latency mode computes the time needed to process a single flow by the controller. We run our experiments on a virtual machine having an Intel(R) Xeon(R) E3-1226 3.3 GHz with 3 cores available and 8 GB of RAM.

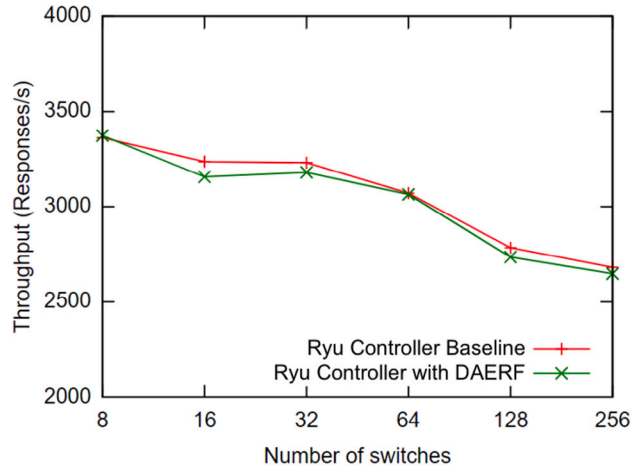

Fig. 8. SDN controller throughput with varying switch count.

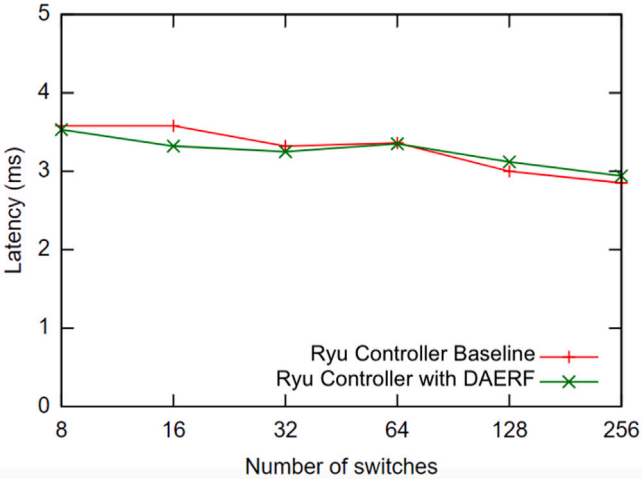

Fig. 9. SDN controller latency with varying switch count.

The operating system is Ubuntu 18.04.2 LTS 64-bit. The controller performance is tested with different numbers of virtual OpenFlow switches emulated by Cbench. The performance of the stand-alone Ryu controller is considered as a baseline for our evaluation and compared to that of a similar controller running DAERF.

4.4.2. Experimental results

Fig. 8 depicts the throughput of a SDN network in two modes: one with a stand-alone Ryu controller, and the other with Ryu running DAERF. We wish to see the effect of the DAERF on the controller's throughput. As can be seen from the Figure, although the throughput tends to decrease with increasing SDN size (as the number of switches increases), this is more attributed to SDN increase in size and not due to DAERF overhead. This is because the same decreasing trend is observed when Ryu is stand-alone, baseline. It is, therefore, clear that DAERF, although securing the SDN network, incurs no throughput degradation of the system.

An interesting observation can be seen from Fig. 9 in that DAERF has almost negligible impact on the controller's latency performance. When we increase the number of connected switches, the latency achieved are between 3 to 4 ms and not so much overhead introduced. This is inline with the observation in Zhu et al. (2019) regarding the Ryu controller latency performance.

5. Conclusion

This paper presented a hybrid combination of an AutoEncoder and Random Forest (DAERF) ML model for intrusion detection in a native SDN environment. We have shown that the proposed DAERF surpasses other previous works with an accuracy of 98% in addition to a decreased amount of time needed for training and execution. From the results accomplished, it is shown that the model is optimistically efficient for real-time intrusion detection.

We have further incorporated our proposed model into a 3-layer intrusion detection framework. This unified attack prevention and mitigation framework has been shown to exhibit good performance when tested on a typical representative dataset. The experimental results have shown that the DAERF_IDS model achieves impressive levels of accuracy, with an average false positive rate of 1.85% and an average detection rate of 98.16%, outperforming previously proposed schemes. Additionally, we have tested the potential overhead such a system would incur. Simulation results revealed that the proposed model has virtually minor overhead when running on a typical SDN controller, making it an attractive model for next generation secure SDN networks.

CRediT authorship contribution statement

Lotfi Mhamdi: Methodology, Project administration, Supervision, Validation, Writing– original draft, Writing – review & editing. **Mohd Mat Isa:** Data curation, Investigation, Methodology, Software, Visualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

Abou El Houda, Z., Hafid, A.S., Khoukhi, L., 2021. A novel machine learning framework for advanced attack detection using sdn. In: 2021 IEEE Global Communications Conference. GLOBECOM, IEEE, pp. 1–6.

Aladaileh, M.A., Anbar, M., Hintaw, A.J., Hasbullah, I.H., Bahashwan, A.A., Al-Sarawi, S., 2022. Renyi joint entropy-based dynamic threshold approach to detect DDoS attacks against SDN controller with various traffic rates. Appl. Sci. 12 (12), 6127.

Ashraf, J., Latif, S., 2014. Handling intrusion and DDoS attacks in software defined networks using machine learning techniques. In: 2014 National Software Engineering Conference. IEEE, pp. 55–60.

Bernico, M., 2018. Deep Learning Quick Reference: Useful Hacks for Training and Optimizing Deep Neural Networks with Tensorflow and Keras. Packt Publishing Ltd.

Binbusayyis, A., Vaiyapuri, T., 2019. Identifying and benchmarking key features for cyber intrusion detection: An ensemble approach. IEEE Access 7, 106495–106513.

Braga, R., Mota, E., Passito, A., 2010. Lightweight DDoS flooding attack detection using NOX/OpenFlow. In: IEEE Local Computer Network Conference. IEEE, pp. 408–415.

Carvalho, R.N., Bordim, J.L., Alchieri, E.A.P., 2019. Entropy-based DoS attack identification in SDN. In: 2019 IEEE International Parallel and Distributed Processing Symposium Workshops. IPDPSW, IEEE, pp. 627–634.

Chang, Y.-P., Li, W., Yang, Z., 2017. Network intrusion detection based on random forest and support vector machine. In: 22017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC). Vol. 01, pp. 635–638, URL https://api.semanticscholar.org/CorpusID:37567415.

Das, S., Liu, Y., Zhang, W., Chandramohan, M., 2015. Semantics-based online malware detection: Towards efficient real-time protection against malware. IEEE Trans. Inf. Forensics Secur. 11 (2), 289–302.

Fan, C., Kaliyamurthy, N.M., Chen, S., Jiang, H., Zhou, Y., Campbell, C., 2021a. Detection of DDoS attacks in software defined networking using entropy. Appl. Sci. 12 (1), 370.

Fan, C., Kaliyamurthy, N.M., Chen, S., Jiang, H., Zhou, Y., Campbell, C., 2021b. Detection of DDoS attacks in software defined networking using entropy. Appl. Sci. 12 (1), 370.

Ferrag, M.A., Maglaras, L., 2019. DeepCoin: A novel deep learning and blockchain-based energy exchange framework for smart grids. IEEE Trans. Eng. Manage. 67 (4), 1285–1297.

Halder, B., Barik, M.S., Mazumdar, C., 2017. A graph based formalism for detecting flow conflicts in software defined network. In: 2017 IEEE International Conference on Advanced Networks and Telecommunications Systems. ANTS, IEEE Press, pp. 1–6. http://dx.doi.org/10.1109/ANTS.2017.8384101.

Hong, R.-F., Horng, S.-C., Lin, S.-S., 2021. Machine learning in cyber security analytics using NSL-KDD dataset. In: 2021 International Conference on Technologies and Applications of Artificial Intelligence. TAAI, pp. 260–265.

Huang, X., Xue, K., Xing, Y., Hu, D., Li, R., Sun, Q., 2020. FSDM: Fast recovery saturation attack detection and mitigation framework in SDN. In: 2020 IEEE 17th International Conference on Mobile Ad Hoc and Sensor Systems. MASS, pp. 329–337. http://dx.doi.org/10.1109/MASS50613.2020.00048.

Ieracitano, C., Adeel, A., Gogate, M., Dashtipour, K., Morabito, F.C., Larijani, H., Raza, A., Hussain, A., 2018. Statistical analysis driven optimized deep learning system for intrusion detection. In: Advances in Brain Inspired Cognitive Systems: 9th International Conference, BICS 2018, Xi'an, China, July 7-8, 2018, Proceedings 9. Springer, pp. 759–769.

Jawaharan, R., Mohan, P.M., Das, T., Gurusamy, M., 2018. Empirical evaluation of SDN controllers using mininet/wireshark and comparison with cbench. In: 2018 27th International Conference on Computer Communication and Networks. ICCCN, pp. 1–2. http://dx.doi.org/10.1109/ICCCN.2018.8487382.

Kaur, K., Singh, J., Ghumman, N.S., 2014. Mininet as software defined networking testing platform. In: International Conference on Communication, Computing & Systems. ICCCS, pp. 139–142.

Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kreutz, D., Ramos, F.M., Verissimo, P.E., Rothenberg, C.E., Azodolmolky, S., Uhlig, S., 2014. Software-defined networking: A comprehensive survey. Proc. IEEE 103 (1), 14–76.

Kumar, P., Tripathi, M., Nehra, A., Conti, M., Lal, C., 2018. SAFETY: Early detection and mitigation of TCP SYN flood utilizing entropy in SDN. IEEE Trans. Netw. Serv. Manag. 15 (4), 1545–1559. http://dx.doi.org/10.1109/TNSM.2018.2861741.

Li, Y., Guo, X., Pang, X., Peng, B., Li, X., Zhang, P., 2020. Performance analysis of floodlight and ryu SDN controllers under mininet simulator. In: 2020 IEEE/CIC International Conference on Communications in China (ICCC Workshops). IEEE, pp. 85–90.

Lim, S., Ha, J., Kim, H., Kim, Y., Yang, S., 2014. A SDN-oriented DDoS blocking scheme for botnet-based attacks. In: 2014 Sixth International Conference on Ubiquitous and Future Networks. ICUFN, pp. 63–68. http://dx.doi.org/10.1109/ICUFN.2014.6876752.

Liu, Y., Zhi, T., Shen, M., Wang, L., Li, Y., Wan, M., 2022. Software-defined DDoS detection with information entropy analysis and optimized deep learning. Future Gener. Comput. Syst. 129, 99–114. http://dx.doi.org/10.1016/j.future.2021.11.009.

McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J., 2008. OpenFlow: Enabling innovation in campus networks. SIGCOMM Comput. Commun. Rev. 38 (2), 69–74. http://dx.doi.org/10.1145/1355734.1355746.

Mousavi, S.M., St-Hilaire, M., 2015. Early detection of DDoS attacks against SDN controllers. In: 2015 International Conference on Computing, Networking and Communications. ICNC, pp. 77–81. http://dx.doi.org/10.1109/ICCNC.2015.7069319.

Nunez, A., Ayoka, J., Islam, M.Z., Ruiz, P., 2023. A brief overview of software-defined networking. arXiv preprint arXiv:2302.00165.

Panigrahi, R., Borah, S., 2018. A detailed analysis of CICIDS2017 dataset for designing intrusion detection systems. Int. J. Eng. Technol. 7 (3.24), 479–482.

Sabeel, U., Heydari, S.S., Mohanka, H., Bendhaou, Y., Elgazzar, K., El-Khatib, K., 2019. Evaluation of deep learning in detecting unknown network attacks. In: 2019 International Conference on Smart Applications, Communications and Networking (SmartNets). IEEE, pp. 1–6.

Sattar, D., Matrawy, A., Adeojo, O., 2016. Adaptive bubble burst (ABB): Mitigating DDoS attacks in software-defined networks. In: 2016 17th International Telecommunications Network Strategy and Planning Symposium (Networks). pp. 50–55. http://dx.doi.org/10.1109/NETWKS.2016.7751152.

Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A., 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization.. ICISSP 1, 108–116.

Shone, N., Ngoc, T.N., Phai, V.D., Shi, Q., 2018. A deep learning approach to network intrusion detection. IEEE Trans. Emerg. Top. Comput. Intell. 2 (1), 41–50.

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. In: 2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft). IEEE, pp. 202–206.

2000. Monit, tildeslash ltd.. URL https://mmonit.com/monit/.

Ujjan, R.M.A., Pervez, Z., Dahal, K., Khan, W.A., Khattak, A.M., Hayat, B., 2021. Entropy based features distribution for anti-ddos model in sdn. Sustainability 13 (3), 1522.

Uppal, H.A.M., Javed, M., Arshad, M., 2014. An overview of intrusion detection system (IDS) along with its commonly used techniques and classifications. Int. J. Comput. Sci. Telecommun. 5 (2), 20–24.

Wang, Y., Hu, T., Tang, G., Xie, J., Lu, J., 2019. SGS: Safe-guard scheme for protecting control plane against DDoS attacks in software-defined networking. IEEE Access 7, 34699–34710.

Yang, L., Song, Y., Gao, S., Hu, A., Xiao, B., 2022. Griffin: Real-time network intrusion detection system via ensemble of autoencoder in SDN. IEEE Trans. Netw. Serv. Manag. 19 (3), 2269–2281. http://dx.doi.org/10.1109/TNSM.2022.3175710.

Ye, J., Cheng, X., Zhu, J., Feng, L., Song, L., 2018. A DDoS attack detection method based on SVM in software defined network. Secur. Commun. Netw. 2018.

Zhang, Y., Chen, X., Jin, L., Wang, X., Guo, D., 2019a. Network intrusion detection: Based on deep hierarchical network and original flow data. IEEE Access 7, 37004–37016.

Zhang, Y., Chen, X., Jin, L., Wang, X., Guo, D., 2019b. Network intrusion detection: Based on deep hierarchical network and original flow data. IEEE Access 7, 37004–37016.

Zhao, Y., Li, Y., Zhang, X., Geng, G., Zhang, W., Sun, Y., 2019. A survey of networking applications applying the software defined networking concept based on machine learning. IEEE Access 7, 95397–95417. http://dx.doi.org/10.1109/ACCESS.2019.2928564.

Zhu, L., Karim, M.M., Sharif, K., Li, F., Du, X., Guizani, M., 2019. SDN controllers: Benchmarking & performance evaluation. arXiv preprint arXiv:1902.04491.

Lotfi Mhamdi (Senior Member, IEEE) received the Master of Philosophy (M.Phil.) degree in computer science from The Hong Kong University of Science and Technology (HKUST), in 2002, and the Ph.D. degree in computer engineering from the Delft University of Technology (TU Delft), The Netherlands, in 2007. He continued his work at TU Delft as a Postdoctoral Researcher, working on various European Union funded research projects. Since 2011, he has been with the School of Electronic and Electrical Engineering, University of Leeds, U.K. His research interests include high-performance networking, Software Defined Networks, Cybersecurity, Internet of Things design, Cloud and Edge Computing convergence and infrastructures design.

Dr. Mhamdi is/was a Technical Program Committee Member of various conferences, including the IEEE International Conference on Communications (ICC), the IEEE GLOBECOM, the IEEE Workshop on High Performance Switching and Routing (HPSR), and the ACM/IEEE International Symposium on Networks-on-Chip (NoCS). He is/was the TPC Co-Chair of the Green Computing, Networking, and Communications Symposium (GCNC 2020), the TPC Co-Chair of GLOBECOM (NGNI Symposium), in 2020 and 2022, IEEE HPSR'24 Workshop Chair and TPC Co-Chair of IEEE ICC'25 IoT & Sensor Networks Symposium. He served as the Chair for the IEEE ComSoc Technical Committee on Communication Switching and Routing (CSR-TC).

Mohd Sani Mat Isa received the B.S. and M.S. degrees in computer networking from Universiti Teknologi Mara, Malaysia, and the Ph.D. degree from University of Leeds, United Kingdom. He is currently working with Malaysian Goverment in managing the Malaysia's Goverment Private Network involving more than 10,000 sites. His research interests include Software Defined Networking (SDN), denial-of-service attacks and security in SDN.