A Survey on the Latest Intrusion Detection Datasets for Software Defined Networking Environments

Harman Yousif Ibrahim Khalid
College of Science, University of Duhok, Kurdistan Region, Iraq
harman.khalid@uod.ac (corresponding author)

Najla Badie Ibrahim Aldabagh
College of Computer Science and Mathematics, University of Mosul, Iraq
najlabadie@uomosul.edu.iq

Received: 16 December 2023 | Revised: 2 January 2024 | Accepted: 10 January 2024

 DOI: https://doi.org/10.48084/etasr.6756

ABSTRACT

Software Defined Networking (SDN) threats make network components vulnerable to cyber-attacks, creating obstacles for new model development that necessitate innovative security countermeasures, like Intrusion Detection Systems (IDSs). The centralized SDN controller, which has global view and control over the whole network and the availability of processing and storing capabilities, makes the deployment of artificial intelligence-based IDS in controllers a hot topic in the research community to resolve security issues. In order to develop effective AI-based IDSs in an SDN environment, there must be a high-quality dataset for training the model to offer effective and accurate attack prediction. There are some intrusion detection datasets used by researchers, but those datasets are either outdated or incompatible with the SDN environment. In this survey, an overview of the published work was conducted using the InSDN dataset from 2020 to 2023. Also, research challenges and future work for further research on IDS issues when deployed in an SDN environment are discussed, particularly when employing machine learning and deep learning models. Moreover, possible solutions for each issue are provided to help the researchers carry out and develop new methods of secure SDN.

Keywords-software defined networking; intrusion detection systems; network security; InSDN; datasets

I. INTRODUCTION

SDN brings significant advantages to network security and has solved many security issues in conventional networks due to its promising features, such as centralized management, bird view, and statistics from forwarding devices to controller. These characteristics improve network security and make it easier to deploy threat detection systems via software applications that make use of open APIs. Despite the benefits offered by the SDN architecture compared to traditional networks, it is vulnerable to cyberattacks and faces new possible threats that do not occur in the traditional networks of today. SDN itself does not have security built-in [1, 2], and one major issue that may hinder the widespread use of SDN is the possibility of new assaults [3]. SDN characteristics, such as network programmability and centralization control introduce new fault and vulnerability, which open the doors for new threats that did not exist before [4, 5]. In [6], seven potential attack vectors against SDNs are listed. Three of these attacks are specific to SDN networks [4, 7–9]. Authors in [10] argue that SDN networks may be more susceptible to malicious traffic than traditional environments due to the decoupling of the control plane and data plane. A security breach in conventional networks causes only minor damage to a small portion of the network, whereas an attack on the SDN controller might have catastrophic consequences for the entire network [7-11]. The attacks target different parts of SDN, as depicted in Figure 1. A countermeasure step to secure SDN is using Intrusion Detection Systems (IDSs) due to their ability to locate and identify malicious activity in the network by examining network traffic in real time [12]. IDS is a tool, which can be either in the form of software or hardware, employed in a network system to observe and assess the behavior of an individual computer or oversee and evaluate the entire network traffic with the aim of distinguishing between legitimate and malicious data flows. As shown in Figure 2, there are two types of IDSs, signature-based and anomaly-based. Signature-based systems are widely used commercially, and malicious traffic can be detected based on the predefined rule. The drawback of this type is that any change performed in the attack signature, even if it is very small, the system will not detect the attack making this type of system unable to detect zero-day attacks [13].

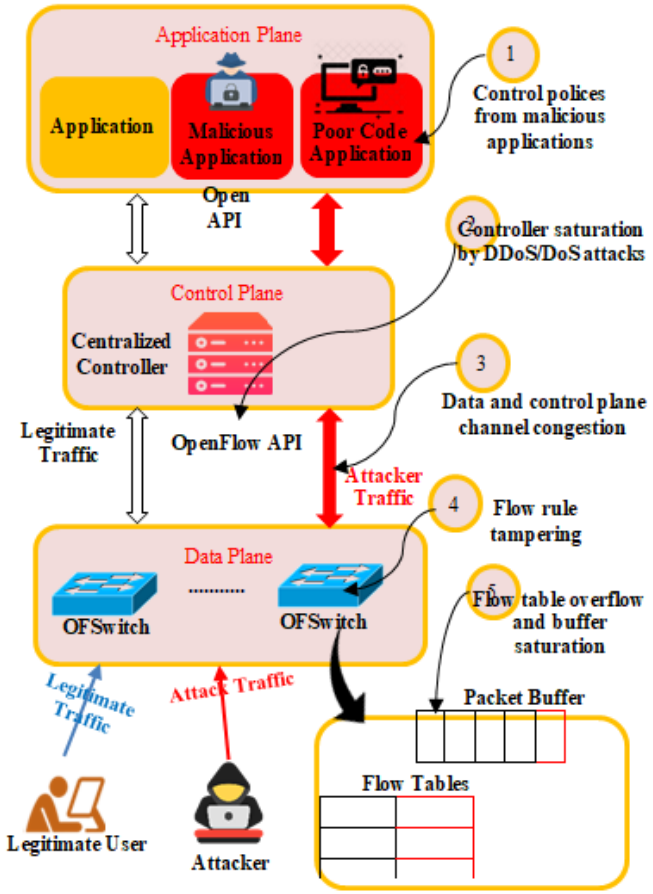

Fig. 1. A taxonomy of SDN security.

Anomaly-based IDSs have attracted significant interest from the academic community due to their capacity to detect new and previously unknown attacks by pinpointing any deviation from the typical traffic pattern. The source of data that is inspected by the IDSs can be the whole data packet, including packet headers and packet payload in packet-based IDS in contrast to the flow-based IDS, which only analyzes the basic information of the communication found in the packet header. Authors in [14] argued that packet content features are not directly accessible in OpenFlow protocol. The amount of data analyzed by flow-based IDSs is less. Therefore, such systems are faster and more computationally efficient than packet-based systems [15]. It is important to note that a flow-based approach is unable to detect attacks embedded in packet payloads. Therefore, to provide a high level of security, both approaches can be implemented together, forming a hybrid method [15]. Deploying IDS in an SDN controller, contrary to the distributed methodology used in traditional networks, not only reduces the expense of purchasing more detection equipment, but also improves detection efficiency due to the centralized view of the controller and the easily obtained statistical details for the IDS application [16]. Fortunately, the centralization of the SDN makes the training of a machine and Deep Learning (DL) based IDSs easier [17] and has become an active area of research [18]. With the use of training data, IDSs possess the ability to automatically learn and recognize patterns from data [20] and the controller's visibility make the deployment and training of AI-based IDSs easier [17]. These IDSs have the ability to modify their behavior based on the changing traffic patterns in the network, making them better equipped to handle emerging security risks [18].

The quality of the detection mechanism depends on the quality of the dataset used for training [20]. One major issue, though, is the lack of high-quality datasets for network traffic and intrusion detection. In various fields, like language translation and computer vision, there are numerous high-publicly accessible quality datasets. The main cause of the intrusion detection domain's lack of publicly available datasets is related to privacy and legal concerns [12]. It is important to note that network data may include sensitive private customer information, and sharing such data with the public would be both unlawful and a breach of privacy. However, anomaly detection in SDN is predicted using public IDS datasets, which are gathered from conventional networks. It is argued [20], that dataset incompatibility with the SDN environment creates false predictions since the framework of SDN is different from the traditional network.

In this survey, an overview of the current approaches that apply DL and Machine Learning (ML) based IDSs in SDN environment is provided. Recent works that used InSDN [21] and the novel SDN-related dataset that was designed to capture unique character attacks in SDN were reviewed.

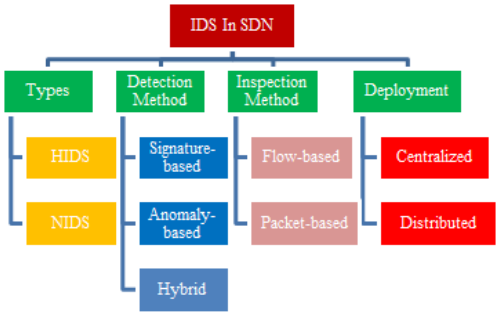

Fig. 2. Taxonomy of IDSs.

II. DATASETS

Using a high-quality dataset for training AI-based IDSs has a great impact on the accuracy of predictions. Only a few datasets related to intrusion detection are publicly available. The primary cause for the lack of publicly available datasets is related to privacy and legal concerns since network data can contain sensitive customer information [22]. However, some available datasets were used for intrusion detection in traditional networks, such as KDDcup99, NSL_KDD, ISCX2012, CICIDS 2017, and CICIDS. KDDcup99 and its revised version, NSL-KDD, are the most popular intrusion detection datasets and are considered benchmarking datasets. However, these datasets are unreliable and outdated, since they were released more than two decades ago [4, 9]. Authors in [16, 17] analyzed both datasets in the SDN context and found that only six (6) features are needed over the 41 available in the SDN environment. Some authors still use those datasets in their

work to predict the performance of new models or to compare them to other datasets, but it is not recommended to use them for real-time detection systems [16]. The ISCX2012 dataset contains only two types of attack: DoS and brute force and HTTP traffic is included as normal traffic, which is not the current standard on the Internet today. Therefore, the dataset is not suitable for modern evaluation. CICIDS 2017 is the updated version of ISCX 2012. It involves a total of 2,830,743 instances, with 19.7% of the total data representing attacks [20]. Those datasets are outdated, have a smaller number of attacks, or are incompatible with the SDN environment [17].

In 2020, a new open-source dataset known as InSDN was made publicly available to be used for the training and evaluation of IDSs within the SDN context [23]. The distribution of the 361,317 total samples in seven attack categories x in the InSDN dataset are shown in Table I. The normal class has 68,424 samples, while the attack class has 292,893 examples [20]. The dataset is split into three groups, including attacks traffic targeting the Metasploitable Server 2 virtual machine as well as the OpenvSwitch (OVS) VM [17]. The normal traffic encompassed many application services, entailing HTTPS, DNS, SSH, FTP, email, etc.

Authors in [7, 15] argue that most of the existing literature views the intrusion detection challenge of SDN as similar to that of a conventional network. Using outdated datasets to train SDN-based IDSs can lead to severe problems, as they are only capable of detecting attacks that exhibit identical behavior in both SDN and traditional networks. The IP-traditional network and SDN are significantly different in their operation. Additionally, the protocols utilized in SDN are not the same as those employed in conventional networks, like OpenFlow [7]. Similarly, new threats have surfaced as a result of the division between the data plane and the control plane. The SDN controller itself or the data communications may be the targets of these assaults.

TABLE I. INSDN DATASET ATTACK CATEGORIES AND DISTRIBUTION

Dataset	Normal traffic/ Attack traffic	Sample number	Total %
Normal	FTP, DNS, HTTPS	68424	68424 (19.9%)
	Attack Category		
Metasploitable 2	DDoS	73529	136743 (39.76%)
	Probe	61757	
	DoS	1145	
	Brute force attack	295	
	R2L	17	
OVS	DoS	52471	138772 (40.34%)
	DDoS	48413	
	Probe	36372	
	Brute force attack	1110	
	Web attack	192	
	Botnet	164	

One example of a novel assault in a SDN environment is a Distributed Denial of Service (DDoS) attack that specifically aims at the SDN controller [24]. The OpenFlow switch will forward the unmatched flow packets it receives to the SDN controller as a packet-in message for additional analysis and handling [20]. An attacker can send various packets with a randomly spoofed destination address even if they do not have to spoof the source address, which is done in traditional networks. All traffic will be forwarded to the controller. The latter will overwhelm it and lead to controller failure, which eventually stops the entire network. Utilizing the InSDN dataset for assessing the effectiveness of anomaly detection models offers more precise outcomes, considering the types of assaults in SDN differ from those often observed in traditional networks. Thus, adopting such a dataset for model evaluation in SDN can serve as a reliable indicator of actual world circumstances. Furthermore, the InSDN dataset is free of any duplicate entries, preventing the learner model from showing a bias towards the most frequently occurring records [12].

III. CURRENT RESEARCH REVIEW

In this survey, existing work is deeply investigated using the InSDN dataset as depicted in Table II. Authors in [25] conducted an analysis of the InSDN dataset, considering all attacks, and presented attack specific feature selection to identify the features that have the greatest impact on anomaly detection to reduce the execution time of the model while maintaining high performance. They argue that because of overfitting and redundant features, a large dataset with many features will take longer for the detection model to execute and might not improve accuracy. To lessen the unbalanced set, they divided the dataset into six new ones, each of which contains normal traffic with a single attack. They performed multiple experiments using the SelectKBest feature selection algorithm to rank the top 10 features for each attack. They found that for all attacks, there were three common features in sequence: duration, Fwd IAT Tot, and Bwd IAT Tot. According to the experiment conducted, duration has the highest impact on all attacks in the InSDN dataset except for the DDoS attack, where Fwd IAT Tot was the most effective. This effectiveness may be attributed to the fact that the victim device is flooded with a huge number of requests as a consequence of the DDoS attack. From the result, it has been observed that both DDoS and probe attacks exhibit the same behavior in that the flow byte feature was dropped suddenly during the attack. The reason behind this behavior is that probe attacks usually scan the target system to discover some information, which results in a very low flow size measured in bytes per second. The normal flow bytes decreased from 0.3332 to zero for DDOS and 0.0001 for probes. However, their methodology of splitting into six datasets means that each attack with a normal sample produces overfitting when detecting attacks, such as botnets and web attacks due to the imbalanced dataset, where the number of samples in a normal traffic sample is much greater than those attacks. Moreover, their work only considered selecting important features for each attack and did not perform any detection classification.

In the same direction, to improve the performance of IDS to detect probe attacks more accurately, the Grey-Wolf Optimizer (GWO) algorithm for feature selection was implemented in [26]. The authors discussed the benefits of feature selection to the overall detection model. They highlighted that feature selection is essential in minimizing computation time, which will make the classifier have high accuracy with optimal

features selected and decrease the dataset size for testing and training. Moreover, for real-time detection, it is easier to extract fewer features, thus decreasing the detection time. They showed that by selecting a subset of 8 features using the LightGBM classifier, accuracy increased to 99.8%, while when using all features, it was 77.3%. Nevertheless, their topology was the same as that of the creator of the dataset [21]. Authors in [10] supported the fact that using many features could be useful in detection accuracy, but it could lead to issues, such as increased model complexity and training costs. Focus is given on various attacks and the Hierarchical Multi-Class (HMC) architecture is proposed to address the imbalance problem in the InSDN dataset and improve the performance of minority classes, like BFA, botnet, and web attacks. To boost the amount of minority class samples, they use SMOTE sampling. To detect abnormal traffic in SDN, they used many ML and DL models, namely K-Nearest Neighbors (KNN), Naïve Bayes (NB), Decision Tree (DT), Random Forest (RF), Adaboost, Bagging, Radial Basis Function Support Vector Machine (RBF-SVM), Linear Support Vector Machine (lin-SVM), Multilayer Perceptron (MLP), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU). Through the conducted experiments, they showed that DDoS, DoS, and probes had good identification performance compared to other attacks because of their majority in the InSDN dataset. At each stage, they used binary classification between the top major class, for instance, the normal traffic class, and the remaining classes in the dataset. In the next stage, after excluding the previous top class, the new top class, such as DDoS, is compared to the rest, and so on. The process of class excluding continues until all classes are filtered by binary classification. Using this method, they improved the detection accuracy of minority classes, like BFA, web attacks, and botnets. However, their method of hierarchical multi-class detection using binary classification at each stage leads to an increase in detection time and computation. Moreover, they implemented and verified their framework for DDoS attacks only.

Some works that have been published have used a hybrid method of ML with DL to improve performance. In [27], an attack detection and mitigation module was proposed that utilized a hybrid model of CNN and Extreme Learning Machine (CNN-ELM) to classify DDoS attacks in an SDN environment. Contrary to previous works, the authors developed a mitigation mechanism along with a detection module. Mitigation is done through IP traceback utilizing a blacklist, which records the abnormal traffic detected by the IDS. Their model detected DDoS attacks by using features extracted from the SDN environment, which were provided by packet-in messages toward the controller as well as the statistics messages provided by OpenFlow switches to the controller. A subset of 12 features from the InSDN dataset were mapped to the OpenFlow switch's extracted features. In addition, they considered four additional features, such as average speed flow, average duration, average packet size, and ratio asymmetric flow. Through the conducted experiments, they demonstrated that using a subset of 12 features not only increased accuracy, but also reduced test time. However, this methodology creates overhead in the controller since every packet-in message, which will not be effective during a DDoS attack, should be checked. Moreover, there was no clear description of how features extracted from packet-in messages. Also, their methodology and the manually created four features were not verified. Similarly, authors in [28], proposed the Deep Convolutional Neural Network (DCNNs) to detect DDoS in SDN. They suggest similar detection and mitigation mechanisms but they used only the features provided by the flow table through OpenFlow statistics messages, and those messages were periodically sent to the controller for anomaly detection. They mentioned that only 12 features of InSDN were mapped to the extracted information from the OpenFlow switch. They argue that the existing solutions engaging a large number of features for ML or DL require more functions to extract those features, which create network congestion and latency. On the other hand, utilizing a small and limited number of features does not provide reliable attack detection. However, in practical implementation, they employed 78 features for training, not only 12, which makes it difficult to map the basic features provided by OpenFlow switches to this huge number. Furthermore, their methodology requires every packet-in message to be checked by the controller, as well as to periodically request statistics from the switch to create overhead in the controller. Similar to previous work, there was no clear description of how the features were extracted from packet-in, while, also, they were not verified.

Authors in [29] examined many DL models, such as the GRU, LSTM, and RNNs, in order to develop an IDS capable of detecting DDoS attacks. They selected 48 features from the InSDN dataset using the framework method described in [30] to acquire the specific features related to SDN. LSTM provided the best accuracy, but in terms of training time, RNN was the most optimal. Nonetheless, they did not consider the imbalanced dataset caused by minority classes when deploying multi-class detection to prevent overfitting. Additionally, there was no real-time classification; in their work, they only considered analyzing DL models. In the same direction, authors in [18] compared several ML models that have high performance with less execution time, such as DT, RF, and Adaboost, to determine the best candidate for the development of ML-based IDS. They revealed that using the SelectKBest function from scikit-learn, which selects relevant features, reduced the number of features from 83 to 7 without significantly degrading performance, while the execution time was reduced rapidly from 10 to 0.5 s. However, as in [29], the authors did consider an imbalanced dataset when using multi-class detection to prevent overfitting.

Authors in [7] propose a hybrid DL approach that combines CNN with RF, KNN, and SVM to classify network traffic. While the aforementioned ML techniques handled the categorization problem, authors employed CNN to extract more complex representations of the data attributes. Most research fails to take into account the impact of overfitting when putting models into practice, which results in poor detection of zero-day attacks. For this reason, a regularization method called SD-Reg was implemented to deal with the overfitting issue. The authors claimed that the imbalance in the InSDN dataset was caused by the insufficient amount of samples for U2R, Web,

and botnet assault. Some attackers always focus on those attacks since the detection models have false predictions about them, or other researchers may ignore them. Thus, in the training multi-classification process, they employed a combination of the oversampling (SMOTE) and undersampling approaches to eliminate randomly selected samples from the majority class and duplicate samples from the minority class. After performing several experiments, they showed that, using softmax with the SD-Reg regularization technique combined with CNN performed better than regularizations L1 and L2. In the next experiment, they replaced softmax with ML techniques, such as SVM, KNN, and RF to work with SD-Reg regularization and for 48 features, compared to single CNN models, CNN-SVM, CNN-KNN, and CNN-RF produced better results for binary and multi-classification. For detecting unknown attacks, for 9 attacks in the dataset, they conducted several experiments with one attack removed from the training set each time and then used for testing, and so on, for all attacks.

Authors in [20] presented a DL model based on LSTM and an autoencoder to detect DDoS attacks in SDN, with a limited number of features to create a lightweight approach to reduce the overhead of applying the detection model. They used Information Gain (IG) and RF algorithms for feature selection to analyze the most relevant feature to the DDoS attack. They utilizedd the same dataset, once with 48 features and once with only 10 features, and they showed that the accuracy does not decrease greately. Their mechanisms include the flow collection and extraction module, which uses OpenFlow statistics messages to get necessary information from switch to controller periodically following a fixed time interval. The authors utilized CICFLOWMETER, which is a tool used for datasets, such as InSDN, CICIDS 2017, and CICIDS 2018, which extract 83 features from traffic flows. They argue that not every CICFLOWMETER feature could be extracted for usage in an SDN setting. Only through OpenFlow calls can the SDN controller obtain statistical data from OpenFlow switches, including flow duration, packet count, and byte count. Thus, they employ the methodology of [30] to identify subfeatures that may be readily retrieved, either directly from the SDN controller, or by computing flow statistics, like the standard deviation, mean, minimum, and maximum of flow features. They conducted experiments and feature selection on three different flow based datasets. It was observed that the subset of selected features in the InSDN dataset is different from those in CICIDS 2017 and CICID 2018, while many common features were found between CICID 2017 and CICIDS 2018 because they are from conventional environment. Traditional networks and SDN platforms are not comparable due to their distinct features and functionalities. Furthermore, the identification of features varies as well as their prominence inside each network. For instance, in a traditional network, flow duration refers to the length of time for which the connection between a source host and a destination host is active. In SDN, flow duration refers to the amount of time that a flow entry remains in the flow table of a switch. Hence, the "duration" attribute in SDN is closely associated with DDoS attacks, as these attacks involve the malicious flow remaining in the switch flow table for a long time. They validated their claim and demonstrated how performance decreases significantly when training with one dataset and testing with another. This validation proves that other datasets that were collected in traditional networks need to be carefully deployed in the SDN network.

Authors in [31] developed an IDS using a hybrid model utilizing an LSTM and CNN combination to extract temporal and spatial information from input data. The accuracy was 96.32% for multi-classification in the InSDN dataset. To overcome the overfitting problem, they used two regularization methods: L2 Reg and dropout. They highlighted that to improve the performance of CNN and detect new intrusions, the overfitting must be reduced to increase accuracy. They used attack samples in testing that were different from those in training. Despite its high accuracy, the high false alarm percentage of the hybrid model could prevent its deployment in the production system. Moreover, the authors stated that their model failed to provide an acceptable result for detecting new attacks. Authors in [13] investigated a methodology to solve the problem of unlabeled and unbalanced dataset. For anomaly identification, they suggest a hybrid strategy based on an LSTM autoencoder and One-Class Support Vector Machine (OC-SVM). They used unsupervised training to solve the problem of an unbalanced dataset by training with a normal class. When there are anomalies, the model produces a significant error because it is unable to identify and rebuild anomaly instances. A threshold known as reconstruct error was employed to distinguish between normal and anomalous data. The shortcoming of OC-SVM, its low capability to work with high-dimension datasets, is solved by combining it with the LSTM autoencoder. The LSTM autoencoder model's data output is reduced to a smaller dimension and then trained using the OC-SVM algorithm to enhance the classification performance. The outcomes of the experiment show that the suggested model provides a higher detection rate. However, they used binary classification with approximately 57,000 normal samples and randomly selected 46,000 samples from all attacks. Combining all attacks in one category with a small number of samples does not provide accurate attack detection, since all attacks have different effective attributes. Moreover, they removed socket features, such as source IP, destination IP, flow ID, etc., only from the dataset to avoid the overfitting problem. Using a large number of features does not solve the issue; in fact, it increases the training and execution time, making the real time implementation impossible.

Authors in [11] introduced a lightweight supervised learning model to identify DDoS attacks targeting SDN controllers using only one feature of fluctuation of flow, which is the count of packet-in messages to the controller in a fixed time slice and for many consecutive times to avoid the behavior of a normal burst. They created their own dataset for the proposed system, but for testing and training their model, they used the InSDN dataset. The idea behind using only one feature is that it will be easier to obtain while it consumes less time and resources for training and real-time prediction. They implemented a multiple ML model with seven selected features of InSDN, which were flow-id, protocol, timestamp, flow-pkt/s, bwd-pkt/s, pkt-len-mean, init-bwd, and win-byts. The conducted experiment shows BT and KNN were the best in terms of accuracy, while in terms of accuracy and training time,

CPU utilization, and decision time, KNN was the most optimal. They tested their work using their own dataset and obtained an accuracy of 99.4% with BT using one feature. It is argued that employing many features will lead to either higher performance or overfitting for some models. However, they did not mention the methodology of feature selection and some of the selected features, such as flow-id and timestamp, which can affect the learning process during model training, leading to overfitting were irrelevant. What is more, continuous checking of the count of packet-in creates a load to the controller, and the methodology of time slices causes a delay in decision time. Ultimately, the technique loses potential because it assesses just one feature to train the intrusion system.

A deep evaluation and analysis was performed in [17], considering IDS implementation in the SDN network and relevant datasets. The author claims that there is a lack of research on IDS in SDN environments. Furthermore, most of the current publication views the IDS issue in SDN similar to that in conventional networks. Also, a significant number of them depend on datasets that were created for conventional networks. They presented novel evaluation work on the InSDN dataset, such as the classification of single/multiple attacks coming from different/same source of data, namely the OVS dataset and Metasploitable 2. Unlike other published research, this one considers training and testing using a single data source. In addition, the importance of the AUC metric for imbalanced dataset classification is highlighted. It was also highlighted that the InSDN dataset is a high quality dataset suitable for IDS in SDN and consists of identical attacks originating from several sources, distinguishing it from former datasets. Other researchers overlooked this particular point. The author conducted several experiments analyzing a new dataset; the same source of data was used for training and testing of single attack detection, and the results were quite satisfactory. However, when conducting the experiment again with different sources for testing and training for single attack detection, the results were good except for BFA and DoS, whose scores were lower in the case of metasploitable server 2 used for training and OVS for testing. It was mentioned that the reason is their limited number of metasploitable server, which affects training. The same problem arose for the probe when the training data source was OVS and testing was conducted on Metasploitable Server 2. Moreover, the experimental multi-classification detection results with the same source for testing and training were exceptional, but for different sources, there was degradation in the performance, due to the existence of heavily populated classes, such as DDoS. It is concluded that the existing technique must be shifted to address the issue of diverse data sources. U2R, botnet, and web attacks were excluded from the analysis due to the limited sample number.

In another direction, some researchers developed ensemble mechanisms to improve performance. Voting algorithms are characterized by low errors and overfitting. When combining multiple classifiers, they provide higher accuracy than a single classifier. By combining NB, KNN, DT, and ET in the V-NKDE ensemble classifier model, authors in [3] created a technique for DDoS detection and mitigation. The voting classifier's concept is to combine many ML algorithms and predict the class label using either an average prediction probability or a majority vote. Furthermore, their systems consist of a collaborative module that notifies other controllers about the attacks, a classifier module for detection, and a mitigation module for blocking attacker ports. The reason behind the collaborative module is that after blocking attacks, there is a vast number of malicious flows in flow tables. These flows are useless, but they waste the memory of the switch. Collaborative modules clear flow tables from such flows. They implement a data collector unit in the controller, which periodically receives OpenFlow statistics messages from OpenFlow switches to obtain details about flow in the flow table. For attack detection, five tuple features in SDN have been considered. To evaluate their model, they used 48 features of InSDN for multi-class detection, with 99.84% accuracy. The experiment showed that the real-time traffic classification accuracy was 99.1% with 0.002 False Positive Rate (FPR).

IV. DISCUSSION

In this survey, a deep analysis was conducted for research employing the InSDN dataset in the SDN environment. After the InSDN dataset was published in 2020, many researchers utilized this dataset during training and testing their models. Among those works, 16 papers were selected. Most of the existing works using InSDN reduced the number of features as follows: 7 of the 16 works used less than 20 features, 6 of them used 48 features [30], and two others used variations in the number of features. The motive behind employing less features was to eliminate overfitting and provide a lightweight intrusion detection system. IG and SelectKBest were the most used feature selection algorithms. Due to the importance of the SDN controller and the fact that it is an attractive target for DDoS attacks, as depicted in Digure 3, seven among the 16 existing solutions performed binary classification and four of them were specific for DDoS detection, one for probe attack detection, and the remaining for classification between normal and abnormal traffic. Various classifiers have been used in the literature and it has been shown that RF was the most optimal. Moreover, LSTM and CNN were often utilized in hybrid classifiers, in combination with CNN achieving good results in image processing due to its power of learning from spatial features, while the power of LSTM resides in its ability to learn from temporal correlations of network traffic that generate times series data [32]. Additionally, LSTM performs better with large datasets [33].

It is clearly illustrated in Figure 4, that among 16 works, four only analyzed the dataset and did not perform any mechanism for attack detection. The remaining 12 works implemented IDS, 5 of them extracted features from OpenFlow statistics messages, only 3 of them used packet-in messages for feature extraction, whereas the remaining did not provide any details about the feature extraction methods used.

Real-time detection is considered an important metric for evaluating the intrusion detection mechanism in the real world and online. However, among those works that deployed intrusion mechanisms, only three of them provided real-time detection. It is clear that real-time detection is not a straightforward process due to the difficulty of getting the required features in real-time and mapping them to dataset features for feeding the classifier model.

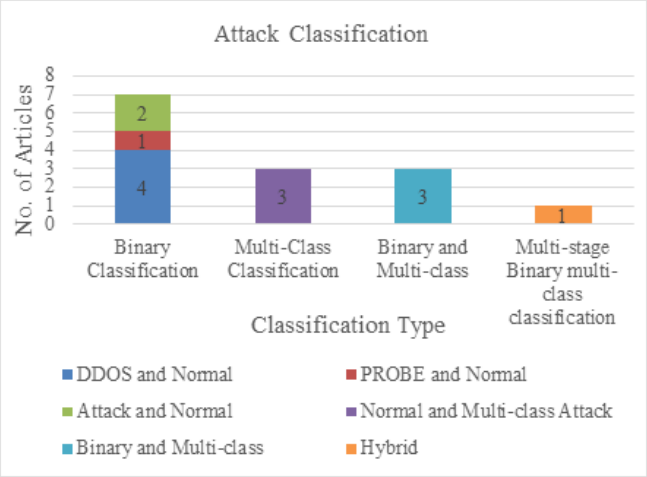

Fig. 3. Binary and multiclass attack distribution in the literature.

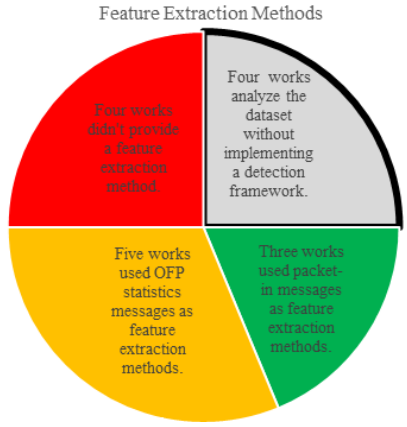

Fig. 4. Feature extraction methods in SDN used in the literature.

The existing models for intrusion detection have several drawbacks. One of them is overfitting, when the model's accuracy is very high during model training, but its performance decreases significantly when tested. This usually happens when we use high-dimensional features. Increasing the number of data samples is an optimal solution, but this method is expensive and constrained by the availability of network data, especially for network traffic. Among the reviewed works, 9 of 16 reduced the feature dimensions using a feature selection algorithm to reduce the effect of overfitting. Removing socket information features, such as timestamp, source IP, destination IP, flow ID, etc. to avoid the overfitting problem as in [12] is not enough. Some other works [4, 27, 30] apply regularization techniques to prevent overfitting. Moreover, some authors used a voting algorithm that was characterized by low error and low overfitting [3]. Another problem is that class imbalance occurs when datasets result from some minority attacks, such as U2R, botnet, and web attacks. Some works did not consider solving this issue because their models performed binary classification between normal and abnormal or normal and attack traffic, namely DDoS and probes [22-24]. Since these are considered majority classes, they did not affect the performance. The effect of class imbalance becomes unavoidable when performing multi-attack classification. Some studies simply neglect minority class attacks, while others use oversampling and undersampling techniques to resolve this issue [10]. Another solution is to use multi-class hierarchical binary classification [10, 15].

Most researchers only focus on providing detection, neglecting mitigation. In the reviewed literature investigated, only four studies provide mitigation. It is difficult and costly to provide a successful protection mechanism, which is why a mitigation mechanism is a favorable option. However, the current work implementing mitigation uses a blocking port mechanism.

V. OPEN ISSUES, CHALLENGES, AND FUTURE RESEARCH DIRECTIONS

In the previous sections, existing solutions for deploying AI-based IDS in an SDN environment were reviewed using the recent novel SDN based dataset. However, in the literature, several issues and challenges were found. In this section, the constraints of the existing methods are identified and potential research concepts to address these limitations are proposed. Additionally, certain future research concerns and challenges are highlighted.

A. Sampling Time Interval for IDS Traffic Monitoring

The IDS must be active all the time to recognize malicious traffic; therefore, IDSs are required to continuously check the traffic and extract features to feed to the detection module. A widespread method for extracting features for AI-based intrusion detection in SDN involves periodically sending OpenFlow statistics messages from the data plane to the controller and extracting the necessary features from the response message. The definition of the time interval to collect flow entries is of great importance. Long time intervals create a delay in the detection of attacks and a reduction in the time available for possible mitigation, allowing an excellent opportunity for the attacker to damage the network. Moreover, it would impose an enormous load on the controller and switch due to the necessity of processing a large number of flows. Conversely, if it is extremely short, the controller will repeatedly engage the detection module, resulting in higher computational costs, increased resource usage by the controller, and increased communication between the controller and switch, leading to bandwidth consumption. The impact of this issue might be insignificant for a network of limited size. However, with larger networks, the problem becomes more severe. There is a need for a mechanism that selects an optimal time interval or a method of invoking the detection mechanism when needed.

B. Accuracy in Real-Time Detection

Previous studies on deploying intrusion IDS in SDN have primarily concentrated on attaining high accuracy through the utilization of novel ML and DL algorithms, or by employing feature selection methods during offline training. There was no clear implementation of real-time detection, and the accuracy of the literature did not reflect reality, as it did not run in real-time. Therefore, detecting attacks in real-time needs to be considered by the research community.

TABLE II. VARIOUS RESEARCH WORKS USED INSDN DATASET

Reference	**[25]**	**[26]**	**[10]**	**[27]**	**[28]**	**[29]**	**[18]**	**[11]**
Year	2021	2022	2022	2022	2023	2021	2023	2022
Controller used	N/A	ONOS	N/A	Ryu	Ryu	N/A	N/A	N/A
Simulation environment	N/A	Mininet	Estinet	Mininet	Mininet	N/A	N/A	N/A
IDS	x	x	x	✓	✓	✓	✓	✓
Dataset	InSDN	InSDN	InSDN	InSDN	InSDN	InSDN	InSDN	InSDNand self-generated
No. of features selected	10	8	20	12	78	48	7	7
Binary / Muti-class	---	Binary	Both	Binary	Binary	Multi-class	Multi-class	Binary
Attack types	---	Probe	All	DDOS	DDOS	All	All	DDOS
Classifier used	Only used feature selection algorithms	LightGBM	Multi-stage Binary Classification	CNN-ELM	DCNN	LSTM, RNN, GRU	DT, RF, Adaboost	Many Classifier. The best were BT and KNN
Feature selection method	SelectKBest	GWO	IG	Manually selected by author + packet-in	Manually selected by author + packet-in	Followed [30]	SelectKBest	N/A
Feature extraction method	Not required	N/A	N/A	OFP_stats message + manually create new feature + packet-in	OFP_stats message + packet-in	N/A	N/A	Count packet-in for self-generated Dataset. Not given for InSDN
Topology	X	✓	✓	X	X	X	X	✓
Framework	X	X	✓	✓	✓	X	X	✓
Real-time detection (deployment)	X	X	✓	X	X	X	X	✓
Accuracy	---	99.80%	96%-99%	99.86%	99.90%	92%	99.80%	Self-generated Dataset: 99.4%. InSDN: 97% - 99%
Evaluation metrics	---	Accuracy, Recall, Precision, F1-score	F1-score	Accuracy, Recall, Precision, F1-score, Test time, Confusion matrix	Accuracy, Recall, Precision, F1-score, Loss rate, Confusion matrix	Accuracy, Recall, Precision, F1-score, Training time, AUC	Accuracy, Recall, Precision, F1-score, Execution time	Accuracy, Recall, Precision, F1-score, Training time, Decision time, CPU utilization
Provide mitigation	X	X	X	✓	✓	X	X	X
Overfitting consideration	Not needed since few features were selected.				N/A	N/A	Not needed since few features were selected.	
Imbalance consideration	N/A	No need to consider because the binary classification between normal and probes	Used Multi-Stage Binary and perform SMOTE for botnet and U2R	No need to consider because the binary classification between normal and DDOS	No need to consider because the binary classification between normal and DDOS	N/A	N/A	No need to consider because the binary classification between normal and DDOS
Remarks	They only analyze the dataset to select relevant features. They did not consider the class imbalance between normal and other minority class attacks, such as botnets and web attacks.	Topology of [21] was considered.	Real-time implementation and verification are performed only for DDOS. Their method will lead to an increase in detection time and computation due to multi-class binary classification.	Four manually created features were not verified. There was no clear methodology for detecting DDOS through packet-in messages.	Overfitting due to the huge number of features. No clear methodology for detecting DDOS through packet-in messages. The mechanism was not verified.	They did not consider imbalanced dataset and overfitting. No real-time classification.	Only analyzed the dataset and did not provide details about IDS implementation. Did not consider an imbalanced dataset.	Did not mention the feature selection methodology and some of the selected features were irrelevant. Continuous checking of the count of packet-ins creates a load on the controller. Time-slice methodology delays decisions.

Reference	**[17]**	**[3]**		**[7]**		**[20]**		**[31]**	**[34]**	**[12]**	**[35]**
Year	2021	2021		2021		2022		2021	2023	2020	2023
Controller used	N/A	Ryu		N/A		Ryu		N/A	N/A	N/A	N/A
Simulation environment	N/A	Mininet		N/A		N/A		N/A	N/A	N/A	N/A
IDS	x	X		X		✓		✓	✓	✓	✓
Dataset	InSDN	InSDN		InSDN		InSDN		InSDN	InSDN	InSDN	InSDN
No. of features selected	N/A	48	5	48	9	48	10	48	48	78	9
Binary / Muti-class	Both	Multi-class	Binary	Both		Binary		Multi-class	Binary then multi-class	Binary	Binary
Attack types	All	All	DDOS	All		DDOS		All	All	All	All
Classifier used	XGBoost	V-NKDE		CNN-RF, CNN-SVM, CNN-KNN		LSTM		CNN-LSTM	LSTM	LSTM-autoencoder + OC-SVM	RF, XGboost
Feature selection method	N/A	Follows [30]	Manually selected	Follow [30]	PCA for 9 features selected	Follows [30]	IG and RF for 10 features selected	Follow [30]		Not used	Follows [7]
Feature extraction method	Not required	N/A	OFP_stats message	Not required		OFP_stats message		N/A	N/A	N/A	OFP_stats message
topology	X	✓		X		X		X	X	X	X
Framework	X	✓		X		✓		X	X	X	X
Real-time detection (deployment)	X	✓		X		X		X	X	X	X
accuracy	N/A	99.84%	99.10%	97% - 99%		99.95%		96.32%	99.50%	90.50%	99.69%
Evaluation metrics	AUC, confusion matrix	Accuracy, Recall, Precision, F1- score, TPR, FPR		Accuracy, Recall, Precision, F1- score, AUC, Confusion matrix		Accuracy, Recall, Precision, F1- score, Execution time, Throughput, Latency		Accuracy, Recall, Precision, F1 score, AUC, Confusion matrix	Accuracy, Detection rate, Precision, F1-score, AUC	Accuracy, Precision, Recall, F1-measure	Accuracy, Precision, Recall, F1-score, Confusion matrix
Provide mitigation	X	✓		X		✓		X	X	X	X
Overfitting consideration	N/A	Voting algorithms are characterized by low error and low overfitting.		Not needed since few features were selected.				Regularization techniques to solve the overfitting problem.		Only removed socket information.	Not needed since only a few features were selected.
Imbalance consideration	N/A	N/A		✓		No need to consider due to the binary classification between normal and DDOS		N/A	N/A	N/A	N/A
Remarks	Only analyzed the dataset and did not perform any detection. Did not mention the number of features selected and using all features without removing features and socket information such as IP, port, and MAC, lead to overfitting. They did not consider an imbalanced dataset.	They did not mention how they trained their model using five tuple features.		Only analyzed the dataset.		Requires periodically getting statistics from the switch to the controller, which creates a load on the controller.		Using a huge number of features makes the feature extraction process very difficult and the detection time very long.	Using a huge number of features makes the detection time very long, and feature extraction creates overhead in an IoT environment that is characterized by limited resource devices. No real-time implementation.	The number of randomly selected samples of all attacks in one category does not provide accurate detection since the attacks have different effective attributes. Used a huge number of features, which created overfitting. Using high-dimensional features increases training and execution time, making real time implementation impossible.	Did not consider an imbalanced class. Methodology of checking OpenFlow statistics message according to limited number of packets in flow instead of fix time window create overhead in the controller.

C. Controller Resource Consumption by IDS

The controller is considered an optimal location for IDS deployment to get the benefit of SDN features as well as exploit the power of the controller [36]. Almost all works implemented their IDS in SDN controllers, but less attention was given to controller resources consumed by IDS. There must be a deep investigation of the controller when network size increases. Integrating IDS as a separate platform and linking it to a controller helps alleviate the stress on the controller [37]. Another solution is to implement a distributed controller or dedicate a specific controller for implementing IDS to migrate excessive traffic for checking to other controllers for processing. However, those methods may increase the time delay for detecting attacks.

D. Consideration of Other Attacks

The nature of the centralized controller of SDN, which is considered the network brain, makes it an attractive target for

DDoS attacks. As a result, many studies of intrusion detection in SDN environments were concerned with DDoS only. The detection procedure needs to be flexible enough to accommodate the additional attack types that the literature has overlooked, such as probe attacks, which have a different methodology of work in SDN from traditional networks. Furthermore, there is a lack of studies considering DDoS attacks targeting OpenFlow switches. Due to the limited memory size of flow tables, attackers can send a burst of forged source packets. The switch will then forward those packets to the controller for decision. After the decision is made by the controller, the rules for those packets are specified and forwarded to the switch. The flow tables in the OpenFlow switch will be unable to store all the fake flow rules. Consequently, the flow table will soon fill up, and the transmission of legitimate traffic will stop.

E. Attack Early Detection

Many research works have developed different high accuracy models, including hybrid, ensemble, etc. These complex models require additional time to detect attacks [38]. Quick identification of an intrusion is crucial, as it enables the initiation of mitigation actions at an earlier stage.

F. Other Methods for Attack Mitigation

The existing solution for attack mitigation in the literature is blocking the attacker port, ignoring the impact the attack caused, and still exist in the network. Performance can be harmed by blocking assaults without taking into account the malicious flow entries that are stored in the switch flow table. When attacks start, there are some useless flow rules installed based on the attacker's behavior, which consume switch resources until they are removed. In addition to the blocking mechanism, there must be other methods used to store the attacker's behavior, which might be helpful in the future for analysis or reference. In this direction, instead of blocking the attacker port, redirecting the malicious traffic to honeypot or mirroring the flow to a deep packet inspector to further analyze the flow is a probable suitable action.

G. Socket Information and Overfitting

Using socket information, such as IP, port, MAC, etc. as direct features [10], leads to overfitting, which has an effect on model prediction. However, it is worth noting that it is a good future direction for researchers employing entropy to measure the distribution of those values for model training or creating other features, namely speed of source IP, standard deviation of flow packets, and standard deviation of flow bytes.

VI. CONCLUSION

SDN features, like flexibility, programmable networks, and dynamic management, successfully resolve the drawbacks of the former network. However, security issues arise because SDN lacks built-in security, and their architecture produces additional security concerns due to the decoupling of data and controller planes. Therefore, implementing intrusion detection in SDN has become a hot topic for the research community to consider in SDN security issues. In this survey, an emphasis was placed on the SDN architecture as a suitable platform for deploying AI-based IDSs to monitor networks and detect threats. Since the efficiency of AI-based intrusion detection depends on the quality of the dataset, a review of InSDN, a new SDN dataset that was collected in an SDN environment, was conducted. Moreover, various research works that used this dataset for intrusion detection development were outlined while their strengths and weaknesses were highlighted. Related research challenges and issues were briefly analyzed and examined. In addition, hypotheses for solving some of those open challenges are provided. There is a strong belief that this survey will help and guide the researchers who aim to develop AI-based IDS solutions in the SDN context.

REFERENCES

[1] L. Kou, S. Ding, T. Wu, W. Dong, and Y. Yin, "An Intrusion Detection Model for Drone Communication Network in SDN Environment," *Drones*, vol. 6, no. 11, Nov. 2022, Art. no. 342, https://doi.org/10.3390/drones6110342.

[2] H. Y. I. Khalid, P. M. Ismael, and A. B. Al-Khalil, "Efficient Mechanism for Securing Software Defined Network against Arp Spoofing Attack," *The Journal of Duhok University*, vol. 22, no. 1, pp. 124–131, Nov. 2019, https://doi.org/10.26682/sjuod.2019.22.1.14.

[3] O. E. Tayfour and M. N. Marsono, "Collaborative detection and mitigation of DDoS in software-defined networks," *The Journal of Supercomputing*, vol. 77, no. 11, pp. 13166–13190, Nov. 2021, https://doi.org/10.1007/s11227-021-03782-9.

[4] 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*, M. Alazab and M. Tang, Eds. New York, NY, USA: Springer, 2019, pp. 175–195.

[5] H. Y. Ibrahim, P. M. Ismael, A. A. Albabawat, and A. B. Al-Khalil, "A Secure Mechanism to Prevent ARP Spoofing and ARP Broadcasting in SDN," in *International Conference on Computer Science and Software Engineering*, Duhok, Iraq, Apr. 2020, pp. 13–19, https://doi.org/10.1109/CSASE48920.2020.9142092.

[6] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015, https://doi.org/10.1109/JPROC.2014.2371999.

[7] M. S. ElSayed, N.-A. Le-Khac, M. A. Albahar, and A. Jurcut, "A novel hybrid model for intrusion detection systems in SDNs based on CNN and a new regularization technique," *Journal of Network and Computer Applications*, vol. 191, Oct. 2021, Art. no. 103160, https://doi.org/10.1016/j.jnca.2021.103160.

[8] G. Logeswari, S. Bose, and T. Anitha, "An Intrusion Detection System for SDN Using Machine Learning," *Intelligent Automation & Soft Computing*, vol. 35, no. 1, pp. 867–880, 2023, https://doi.org/10.32604/iasc.2023.026769.

[9] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for Network Intrusion Detection in Software Defined Networking," in *International Conference on Wireless Networks and Mobile Communications*, Fez, Morocco, Oct. 2016, pp. 258–263, https://doi.org/10.1109/WINCOM.2016.7777224.

[10] H.-M. Chuang, F. Liu, and C.-H. Tsai, "Early Detection of Abnormal Attacks in Software-Defined Networking Using Machine Learning Approaches," *Symmetry*, vol. 14, no. 6, Jun. 2022, Art. no. 1178, https://doi.org/10.3390/sym14061178.

[11] S. Wang *et al.*, "Detecting flooding DDoS attacks in software defined networks using supervised learning techniques," *Engineering Science and Technology, an International Journal*, vol. 35, Nov. 2022, Art. no. 101176, https://doi.org/10.1016/j.jestch.2022.101176.

[12] M. Said Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Network Anomaly Detection Using LSTM Based Autoencoder," in *16th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, Alicante, Spain, Nov. 2020, pp. 37–45, https://doi.org/10.1145/3416013.3426457.

[13] N. A. Alsharif, S. Mishra, and M. Alshehri, "IDS in IoT using Machine Learning and Blockchain," *Engineering, Technology & Applied Science Research*, vol. 13, no. 4, pp. 11197–11203, Aug. 2023, https://doi.org/10.48084/etasr.5992.

[14] A. D. Althobiti, R. M. Almohayawi, and O. O. Bamsag, "Machine Learning approach to Secure Software Defined Network: Machine Learning and Artificial Intelligence," in *4th International Conference on Future Networks and Distributed Systems*, Saint Petersburg, Russian, Nov. 2020, pp. 1–8, https://doi.org/10.1145/3440749.3442597.

[15] M. Latah and L. Toker, "An efficient flow-based multi-level hybrid intrusion detection system for software-defined networks," *CCF Transactions on Networking*, vol. 3, no. 3, pp. 261–271, Dec. 2020, https://doi.org/10.1007/s42045-020-00040-z.

[16] E. M. Zeleke, H. M. Melaku, and F. G. Mengistu, "Efficient Intrusion Detection System for SDN Orchestrated Internet of Things," *Journal of Computer Networks and Communications*, vol. 2021, Nov. 2021, Art. no. e5593214, https://doi.org/10.1155/2021/5593214.

[17] Q.-V. Dang, "Intrusion Detection in Software-Defined Networks," in *Future Data and Security Engineering*, Nov. 2021, pp. 356–371, https://doi.org/10.1007/978-3-030-91387-8_23.

[18] A. Mzibri, R. Benaini, and M. B. Mamoun, "Case Study on the Performance of ML-Based Network Intrusion Detection Systems in SDN," in *International Conference on Networked Systems*, Benguerir, Morocco, Dec. 2023, pp. 90–95, https://doi.org/10.1007/978-3-031-37765-5_7.

[19] S. Singh and S. Banerjee, "Machine Learning Mechanisms for Network Anomaly Detection System: A Review," in *International Conference on Communication and Signal Processing*, Chennai, India, Jul. 2020, pp. 976–980, https://doi.org/10.1109/ICCSP48568.2020.9182197.

[20] M. S. E. Sayed, N.-A. Le-Khac, M. A. Azer, and A. D. Jurcut, "A Flow-Based Anomaly Detection Approach With Feature Selection Method Against DDoS Attacks in SDNs," *IEEE Transactions on Cognitive Communications and Networking*, vol. 8, no. 4, pp. 1862–1880, Sep. 2022, https://doi.org/10.1109/TCCN.2022.3186331.

[21] M. S. Elsayed, N.-A. Le-Khac, and A. D. Jurcut, "InSDN: A Novel SDN Intrusion Dataset," *IEEE Access*, vol. 8, pp. 165263–165284, 2020, https://doi.org/10.1109/ACCESS.2020.3022633.

[22] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, pp. 147–167, Sep. 2019, https://doi.org/10.1016/j.cose.2019.06.005.

[23] "Index of /datasets/SDN." https://aseados.ucd.ie/datasets/SDN/.

[24] M. H. H. Khairi, S. H. S. Ariffin, N. M. A. Latiff, A. S. Abdullah, and M. K. Hassan, "A Review of Anomaly Detection Techniques and Distributed Denial of Service (DDoS) on Software Defined Network (SDN)," *Engineering, Technology & Applied Science Research*, vol. 8, no. 2, pp. 2724–2730, Apr. 2018, https://doi.org/10.48084/etasr.1840.

[25] N. Abbas, Y. Nasser, M. Shehab, and S. Sharafeddine, "Attack-Specific Feature Selection for Anomaly Detection in Software-Defined Networks," in *3rd IEEE Middle East and North Africa COMMunications Conference*, Agadir, Morocco, Dec. 2021, pp. 142–146, https://doi.org/10.1109/MENACOMM50742.2021.9678279.

[26] A. Almazyad, L. Halman, and A. Alsaeed, "Probe Attack Detection Using an Improved Intrusion Detection System," *Computers, Materials & Continua*, vol. 74, no. 3, pp. 4769–4784, 2023, https://doi.org/10.32604/cmc.2023.033382.

[27] J. Wang and L. Wang, "SDN-Defend: A Lightweight Online Attack Detection and Mitigation System for DDoS Attacks in SDN," *Sensors*, vol. 22, no. 21, Jan. 2022, Art. no. 8287, https://doi.org/10.3390/s22218287.

[28] V. Hnamte and J. Hussain, "An efficient DDoS attack detection mechanism in SDN environment," *International Journal of Information Technology*, vol. 15, no. 5, pp. 2623–2636, Jun. 2023, https://doi.org/10.1007/s41870-023-01332-5.

[29] A. S. Alshra'a, A. Farhat, and J. Seitz, "Deep Learning Algorithms for Detecting Denial of Service Attacks in Software-Defined Networks," *Procedia Computer Science*, vol. 191, pp. 254–263, Jan. 2021, https://doi.org/10.1016/j.procs.2021.07.032.

[30] P. Krishnan, S. Duttagupta, and K. Achuthan, "VARMAN: Multi-plane security framework for software defined networks," *Computer Communications*, vol. 148, pp. 215–239, Dec. 2019, https://doi.org/10.1016/j.comcom.2019.09.014.

[31] M. Abdallah, N. An Le Khac, H. Jahromi, and A. Delia Jurcut, "A Hybrid CNN-LSTM Based Approach for Anomaly Detection Systems in SDNs," in *16th International Conference on Availability, Reliability and Security*, Vienna, Austria, Aug. 2021, pp. 1–7, https://doi.org/10.1145/3465481.3469190.

[32] O. M. Ahmed, L. M. Haji, A. M. Ahmed, and N. M. Salih, "Bitcoin Price Prediction using the Hybrid Convolutional Recurrent Model Architecture," *Engineering, Technology & Applied Science Research*, vol. 13, no. 5, pp. 11735–11738, Oct. 2023, https://doi.org/10.48084/etasr.6223.

[33] R. Alsulami, B. Alqarni, R. Alshomrani, F. Mashat, and T. Gazdar, "IoT Protocol-Enabled IDS based on Machine Learning," *Engineering, Technology & Applied Science Research*, vol. 13, no. 6, pp. 12373–12380, Dec. 2023, https://doi.org/10.48084/etasr.6421.

[34] R. A. Elsayed, R. A. Hamada, M. I. Abdalla, and S. A. Elsaid, "Securing IoT and SDN systems using deep-learning based automatic intrusion detection," *Ain Shams Engineering Journal*, vol. 14, no. 10, Oct. 2023, Art. no. 102211, https://doi.org/10.1016/j.asej.2023.102211.

[35] M. S. Towhid and N. Shahriar, "Early Detection of Intrusion in SDN," in *IEEE/IFIP Network Operations and Management Symposium*, Miami, FL, USA, Dec. 2023, pp. 1–6, https://doi.org/10.1109/NOMS56928.2023.10154272.

[36] A. Abubakar and B. Pranggono, "Machine learning based intrusion detection system for software defined networks," in *Seventh International Conference on Emerging Security Technologies*, Canterbury, UK, Sep. 2017, pp. 138–143, https://doi.org/10.1109/EST.2017.8090413.

[37] S. Kumar *et al.*, "DDoS Detection in SDN using Machine Learning Techniques," *Computers, Materials & Continua*, vol. 71, no. 1, pp. 771–789, 2022, https://doi.org/10.32604/cmc.2022.021669.

[38] A. O. Alzahrani and M. J. F. Alenazi, "Designing a Network Intrusion Detection System Based on Machine Learning for Software Defined Networks," *Future Internet*, vol. 13, no. 5, May 2021, Art. no. 111, https://doi.org/10.3390/fi13050111.