sustainability

MDPI

Article

Detecting DDoS Attacks in Software-Defined Networks Through Feature Selection Methods and Machine Learning Models

Huseyin Polat *, Onur Polat and Aydin Cetin *

Faculty of Technology, Gazi University, Ankara 06500, Turkey; onurpolat@gazi.edu.tr

* Correspondence: polath@gazi.edu.tr (H.P.); acetin@gazi.edu.tr (A.C.)

Received: 19 December 2019; Accepted: 29 January 2020; Published: 1 February 2020

Abstract: Software Defined Networking (SDN) offers several advantages such as manageability, scaling, and improved performance. However, SDN involves specific security problems, especially if its controller is defenseless against Distributed Denial of Service (DDoS) attacks. The process and communication capacity of the controller is overloaded when DDoS attacks occur against the SDN controller. Consequently, as a result of the unnecessary flow produced by the controller for the attack packets, the capacity of the switch flow table becomes full, leading the network performance to decline to a critical threshold. In this study, DDoS attacks in SDN were detected using machine learning-based models. First, specific features were obtained from SDN for the dataset in normal conditions and under DDoS attack traffic. Then, a new dataset was created using feature selection methods on the existing dataset. Feature selection methods were preferred to simplify the models, facilitate their interpretation, and provide a shorter training time. Both datasets, created with and without feature selection methods, were trained and tested with Support Vector Machine (SVM), Naive Bayes (NB), Artificial Neural Network (ANN), and K-Nearest Neighbors (KNN) classification models. The test results showed that the use of the wrapper feature selection with a KNN classifier achieved the highest accuracy rate (98.3%) in DDoS attack detection. The results suggest that machine learning and feature selection algorithms can achieve better results in the detection of DDoS attacks in SDN with promising reductions in processing loads and times.

Keywords: SDN; AI algorithms; feature selection; DDoS attacks

1. Introduction

Traditional network infrastructures have been unable to address certain requirements such as high bandwidth, accessibility, high connection speed, dynamic management, cloud computing, and virtualization. Thus, Software Defined Networking (SDN), which has a flexible, programmable, and dynamic architecture, has emerged as an alternative to traditional networks [1]. SDN architecture consists of control, data, and application planes. Devices, such as switches and routers, are placed on the data plane. This plane is programmed and managed by the control plane [2]. The control plane is responsible for the management of transmission devices placed on the data plane. The controller, which performs as the brain of the network, is located on this plane. Devices on the data plane carry out packet transmission according to the rules set by the controller. The application plane communicates with the devices located on the network infrastructure via the controller.

Devices on the network are programmed by applications using Northbound and Southbound interfaces. The controller communicates with the services working on the application layer by means of the Northbound interface and communicates with the devices on the data layer by means of the Southbound interface. The most popular protocol used in this interface is the OpenFlow protocol [3].

As the control logic has been taken from local devices and become central, SDN is structured from a single spot and dynamically optimized [4]. Although certain security threats are general in computer networks, SDN has brought its own security threats. There are at least seven different threat vectors identified with SDN [5]. The most important threat vector for SDN is Distributed Denial of Service (DDoS) attacks. They can be against the controller in SDN or in the storage capacity of the flow table in the OpenFlow switch.

The controller is exposed to DDoS attacks through the communication line between the controller and the data plane. DDoS attacks direct a large amount of traffic to the OpenFlow switch on the data plane. If packets arriving at the OpenFlow switch do not match with the flow input in the flow table (miss flow), packets are taken into the flow buffer. Then, they are transmitted to the controller with the Packet-In message to write a new rule. In this situation, the sources of the controller (memory, processor, bandwidth, etc.) remain incapable and the network becomes inoperative. In addition, the bandwidth of the communication line between the controller that is exposed to attack traffic and the OpenFlow switch is negatively affected. Therefore, network performance severely declines [6].

The data plane is exposed to DDoS attacks through the flow table located in the network devices. In such DDoS attacks, packets are sent to the switch from unknown sources. The controller writes a rule for these arriving packets and forwards them to the flow table of the switch. After a while, the capacity of switch flow table becomes full. Consequently, new rules cannot be written to the flow table and packets cannot be forwarded. In addition, the incoming packets fall automatically as the buffer capacity becomes full with the attack [7]. OpenFlow switches manage the packets with only the flow entry coming from the controller. Thus, decreasing DDoS attacks in SDN is more difficult when compared to traditional networks.

Machine learning-based approaches can provide more dynamic, more efficient, and smarter solutions for SDN management, security, and optimization. In order to ensure network security, the detection of DDoS attacks is very important for taking the necessary measures in a timely manner. Processing SDN flow data with the machine learning-based DDoS attack detection systems integrated into SDN structures can lead to a self-determining network that is able to learn and act. Moreover, SDN having an integrated machine learning application may be a reference model in forming a secure structure in studies providing for the integration of 5G networks.

This paper consists of five sections. The second section presents a short review on DDoS attacks against the controller and DDoS attacks against the OpenFlow switch, along with SDN architecture and properties and security gaps. In the third section, under Materials and Methods, process steps for applying the feature selection methods and machine learning models, and experimental setup and data collection from SDN are presented in detail. In the fourth section, the experimental results are analyzed and implications are drawn from the findings. In the last section of the study, the concluding remarks of the study are presented.

2. Related Work

We discuss the related studies from two perspectives, namely, DDoS attacks which are against the controller and against the OpenFlow switch.

2.1. DDoS Attacks against the Controller

Control functions are taken from the switch and given to the controller, which is the brain of the network in SDN architecture. Parent level rules are easily applied to the network with the help of the controller. The controller can add new rules to the transmission devices and change the existing rules. It can carry out these changes by communicating with transmission devices via a secure channel through the OpenFlow protocol. Continuity and the unity of data traffic are ensured through this channel. If this secure channel breaks, the connection between the controller and transmission devices breaks.

SDN architecture is the target for DDoS attacks. While the attacker is attacking the SDN network, it has three main targets, as shown in Figure 1: to consume the sources of the controller, to occupy the

bandwidth of the channel between the controller and the switch, and to fill the flow tables in the switch with unnecessary flows. In DDoS attacks against the controller, the attacker sends a large number of packets to the OpenFlow switch via zombie users. It is difficult for the controller to differentiate between traffic sent by the attacker and legal traffic.

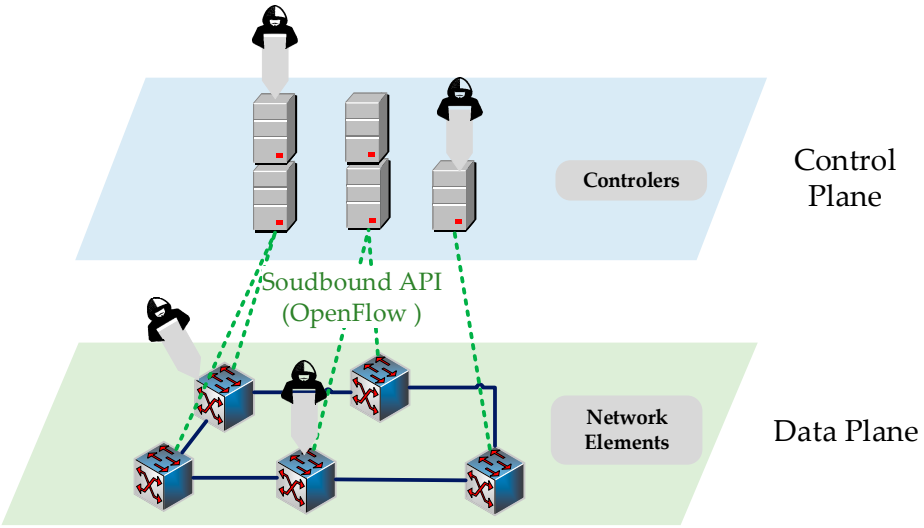

Figure 1. Main targets of the Distributed Denial of Service (DDoS) attacks on Software Defined Networking (SDN).

The OpenFlow switch seeks a match in the flow input by checking the packet header (source port, target port, source IP address, target IP address, etc.). If there is no match, the packet is forwarded to the controller by encapsulating the packet header in the flow request with the OpenFlow protocol (OFPT) PACKET_IN message. Then, the controller responds with the OFPT FLOW_MOD message. This message involves the process to be carried out on the packet and the timeout of the flow in the flow table assigned for the packet [8]. As the number of packets forwarded to the controller increases, the sources of the controller are consumed (bandwidth, memory, and CPU), new flow input for the new legal packets arriving at the network cannot be processed, and the SDN architecture collapses.

Alshamrani et al. [9] claim that the existing mechanisms to prevent DDoS attacks are not effective. Accordingly, they investigated the effect of misbehavior and new flow attacks on SDN. They gathered traffic data from transmission devices on a data plane periodically and then applied machine learning classification algorithms to respond to sudden traffic changes that occurred in the SDN architecture at the moment of attack. Packet_In messages flowing between the controller and transmission devices at the moment of attack were used as a base. Support Vector Machine (SVM), J48, and Naive Bayes (NB) algorithms were employed for classification.

Latah and Toker [10] detected DDoS attacks by measuring the rate of the arriving packet at the moment of attack. When some packets, among the ones coming to the controller, passed the pre-determined threshold, an inspection unit that uses SVM was activated to be able to detect DDoS flooding attacks.

Li et al. [11] proposed a bidirectional Recurrent Neural Network (RNN) model covering each layer of the SDN network structure in order to detect and block DDoS attacks. Although the proposed model was developed for the real-time detection, and the blocking of DDoS attacks has a high accuracy rate, it may not be equally successful in large networks where more than one controller is used. The proposed model may disrupt the synchronized work of controllers and degrade the performance of the network.

2.2. DDoS Attacks against the OpenFlow Switch

The OpenFlow switch and flow table are seen as the main targets, as they include administrative, transmission, and access control information [12]. The attacker first aims to break the functionality of the network by way of physical or virtual unauthorized access to the network. It is impossible for the OpenFlow switch to store all the rules covering each flow as it has a limited storage capacity. As packets come from an unknown address, new rules are needed to be added to the switch. If the attacker sends a large number of packets from an unknown address in a short time, the rules are written by the controller for these packets and forwarded to the flow table. The flow table with limited storage space becomes full in a short time. No space is left in the flow table for a new rule. Thus, transmission of the legal traffic stops. Apart from the flow table, flow cache memory is also a target for DDoS attacks (Figure 1).

Upon receiving the packet from the input port, the switch with a reactive cache memory mechanism forwards this packet to the flow cache memory [13]. For the new arriving packets, a match is searched for the flows in a flow table. If a match is found, the packet is forwarded from the cache memory to the output port. Otherwise, it is forwarded to the controller by means of the control channel and with a Packet_In message. The controller responds with an OFPT_FLOW_MOD message by assigning hard_timeout and idle_timeout, which define the necessary rules for the packet and how long the rules will stay. When the switch receives the rule from the controller, the packet is processed and the rule written by the controller is taken into the transmission table cache memory to have the arriving packets directly processed. This mechanism makes the switch defenseless against DDoS attacks. A large number of packets forwarded to the switch by malicious nodes are taken into cache memory and the response from the controller, which includes the information on the flow rules, is expected [14]. Packets coming from malicious nodes fill the switch buffer and then packets coming from legal users start to drop [15].

Ye et al. [16] gathered data on network traffic from the transmission devices on the data plane by means of the controller. Six-tuple characteristic values related to DDoS attacks from the switch flow table were extracted to detect DDoS attacks by means of SVM. A high detection accuracy rate was reported. However, the test accuracy rate of the Internet Control Message Protocol (ICMP) attack flow was reported as relatively low.

Xue et al. [17] indicated that many security requirements are needed as the SDN controller manages multiple switches in the data plane. They stated that security could not be achieved with existing equipment and software that has not yet adapted to the SDN architecture. In particular, DDoS attack on switches in the data plane pose a great danger for the continuity of the SDN architecture of the network. SVM-optimized C and G parameters by cross-validation-genetic algorithm (CV-GA) were used to detect the DDoS attacks.

Nanda et al. [18] proposed the use of machine learning algorithms, trained on historical network attack data, to identify potential malicious connections and potential targets to minimize security threats. They used C4.5, Bayesian Network (BayesNet), Decision Table (DT), and Naive Bayes algorithms. It was stated that detecting malicious users in the data plane by means of estimations using machine learning algorithms was possible. Ensuring user identification can enable the SDN controller to quickly and effectively write new rules to prevent the attack, which is important for the efficiency and continuity of the network.

Jankowski and Amanowicz [19] employed the machine learning algorithms of Self-Organizing Maps (SOM), Learning Vector Quantization (LVQ1), and their enhanced versions (Multi-pass Self-Organizing Maps (M-SOM), Multi-pass Learning Vector Quantization (M-LVQ1) and Hierarchical Learning Vector Quantization (H-LVQ1)) to detect and monitor malicious activities on the data plane. Promising results were achieved with the H-LVQ1 algorithm compared to SOM, M-SOM, LVQ1, M-LVQ1.

Banerjee and Chakraborty [20] conducted a two-stage study. In the first stage, Naive Bayes and K-Nearest Neighbors (KNN) machine learning algorithms were trained to differentiate between

the attack and normal traffic. Then the attacker was detected using a three-way handshake service. The detected attacker was blocked by creating an Access Control List (ACL).

Mowla et al. [21] proposed Cognitive Switch-based DDoS Sensing and Mitigation in SDN-driven Content Delivery Networks. SVM and Logistic Regression algorithms were used for traffic classification, and this classification initiated the deployment of security rules to the OpenFlow switches, to prevent from new forms of flooding attacks, and to detect and defend against all possible DDoS attacks.

This study focuses on DDoS attacks in SDN architecture and offers machine learning models supported with feature selection methods to detect attacks. Therefore, the aim is to develop DDoS attack detection systems based on a high-fertility rate machine learning for SDN architecture. The detection of the DDoS attack on the SDN controller and the switches located in the data plane are very important for the continuity of the network and the detection of legal traffic at the time of the attack. If attack traffic is detected on the controller, it is easier for the controller to write new rules to the flow table of the switches located in the data plane to prevent the attack. This provides a great advantage for preventing the attack. We propose the use of feature selection methods with machine learning models to detect DDoS attacks. We believe that our approach will make a significant contribution to the effective detection of DDoS attacks in SDN.

3. Materials and Methods

The process steps for applying the feature selection methods and machine learning models are given in Figure 2. Additionally, in this section, how the data is obtained from the dataset, and the features and classes of the dataset are explained. The features in the dataset, created within the scope of the study, were obtained from the main metrics that are vital for the continuity of the SDN architecture. The attack data, obtained as a result of DDoS attacks, were classified using machine learning techniques. Features in the dataset were selected using feature selection methods and the performances of the classifiers on the obtained feature set were examined. The analysis of the application was performed in a Matlab environment.

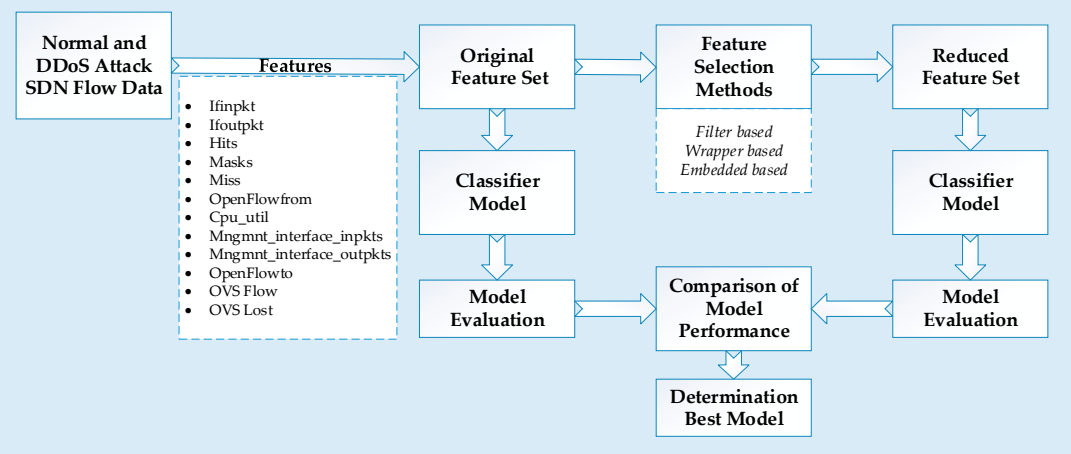

Figure 2. Process steps for applying the feature selection methods and machine learning models.

3.1. Feature Selection Methods and Machine Learning Models

To detect DDoS attacks, machine learning models supported with feature selection methods are used in this study. Machine learning is a method that draws implications from existing data by using mathematical and statistical methods, and makes predictions about the unknown with these implications. In the literature, a variety of machine learning models have been proposed. The taxonomy of models can be summarized with the kernel-based, distance-based, neural network- based and probability-based characteristics. The continuing research suggests that there is no universal best model for all classification tasks. The literature suggests that SVM, KNN, ANN (Artificial Neural

Network), and NB models have shown good performance for solutions of classification problems. In this study, as candidates of their classification group, the performance rate for detecting DDoS attacks from SVM, KNN, ANN, and NB models were investigated.

The datasets used in the training test of machine learning models can contain a large number of features. Although some of these features have a high degree of influence on the classification result, some features may have little or no effect on the classification result. Using features that have little impact on classification can increase time and process costs. The goal is to reduce the features that have a low impact on the classification and to provide the use of highly effective features.

The filter, embedded, and wrapper methods were used as feature selection methods. The filter method mainly focuses on the intrinsic properties of the features, while the wrapper method focuses on the usefulness of features based on the classifier performance. The embedded method tries to use the properties of both the filter and wrapper methods to optimize the performance of a learning algorithm or model. Each feature selection method has different algorithms to achieve their goals. Our classification problem is a binary classification problem and in our dataset there is no missing data. Therefore, the relief filtering algorithm inspired by instance-based learning was selected for filter-based feature selection.

A greedy search-based Sequential Forward Selection (SFS) algorithm was preferred due to its proven success in finding an optimal feature subset among the wrapper-based feature selection methods. Lasso or L1 algorithm was selected as the embedded feature selection method as it adds a penalty against complexity to reduce the degree of overfitting, which improves the optimization performance of the model.

3.1.1. Filter-Based Feature Selection

The filter-based feature selection method contributions to the achievement of the results of the features are calculated using statistical methods. A reduced feature set is created by selecting the best feature set from the rated features, as seen in the process given in Figure 3. In the study, the Relief algorithm, which is one of the filter-based feature selection algorithms, is used. The Relief algorithm calculates the relation of each sample in the dataset with other instances in its class and its distance to different classes. As a result of this calculation, it creates a model and performs the feature selection process [22,23].

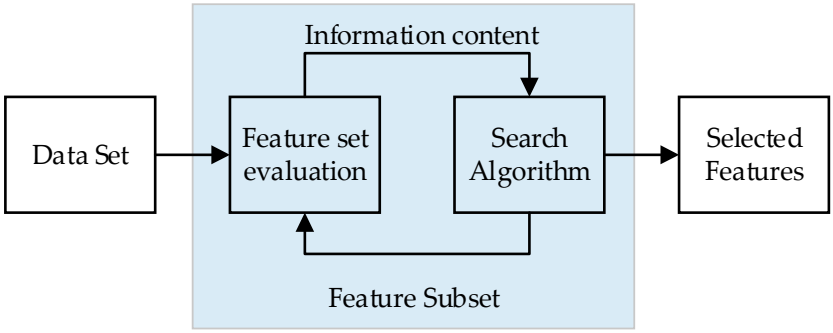

Figure 3. A filter-based feature selection algorithm.

3.1.2. Wrapper-Based Feature Selection

The wrapper method aims to find the most ideal attributes by trying all features in the classification algorithm. When the ideal subset of features is reached, the process is completed and a reduced dataset is created (Figure 4). This may be more successful than statistical methods; however, depending on the classification algorithm, it may be disadvantageous in terms of speed. In the study, the sequential forward floating selection algorithm, which is one of the wrapper-based feature selection algorithms, is used. When selecting a feature, the same direction is used until the classification success reaches a higher value [22,23].

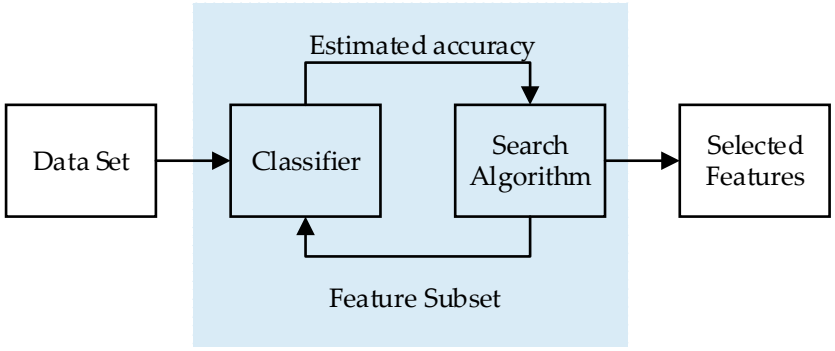

Figure 4. A wrapper-based feature selection algorithm.

3.1.3. Embedded-Based Feature Selection

Embedded-based feature selection methods are feature selection algorithms that are included in the classification algorithm. These algorithms make their selection of features by identifying the features that will best contribute to the accuracy of the model (Figure 5). They have been developed to combine the advantages of filter and wrapper-based methods. In this method, a learning algorithm takes advantage of its variable selection process and simultaneously performs feature selection and classification [22]. In this study, the Lasso algorithm, which is one of the embedded-based feature selection algorithms, is used. An important characteristic of the Lasso algorithm eliminates the weights of the least important features. That is, in this way, it naturally makes the attribute selection and outputs a reduced set of features [22,23].

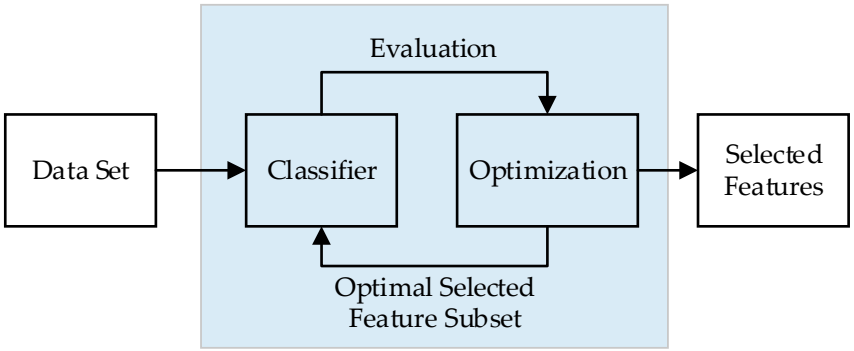

Figure 5. An embedded-based feature selection algorithm.

3.1.4. Support Vector Machine (SVM)

SVM is a supervised learning model employed in the learning field, which uses classification and regression analysis in defining and analyzing obtained patterns. The working principle of SVM is based on predicting the most appropriate decision function to separate between two classes. In other words, it is based on defining the hyper-plane that can separate two classes. We used SVM with Cubic kernels.

3.1.5. Naive Bayes

NB classification aims to detect the class, namely the category, of the data entered into the system with a series of calculations defined in accordance with the principles of probability [23]. In NB classification, a certain amount of training data is entered into the system. The data entered for training has to have a class. Test data entered into the system with probability processes carried out on the training data is processed. This process is done according to previously obtained probability values and then the class of the given test data is detected. The more the number of training data, the more accurate it is to detect the real category of the test data. We used NB with Gaussian kernels.

3.1.6. Artificial Neural Network (ANN)

ANN is a strong classifying tool based on the artificial neuron model. It uses neurons as the most basic building block. Artificial neurons are designed to behave similarly to biological neurons. The ANN architecture created in this study contains 12 input units and 10 neurons in a single hidden layer. It should be noted that the number of inputs and the number of neurons in a single hidden layer change according to the chosen filter methods used in feature selection.

3.1.7. K-Nearest Neighbors

The KNN algorithm is a simple, easily applicable, and supervised machine learning algorithm that can be used for solving both classification and regression problems. When new data comes in, it determines the class of the new data by looking at its nearest K neighbors. Manhattan, Minkowski, and Euclidean distance functions are used for the distance between two data. The Euclidean distance function was used in this study. The similarity between the sample to be classified and the samples found in the classes was detected. When new data was encountered, the distance of this data to the data in the training set was calculated individually by using the Euclidean function. Then, the classification set was created by selecting the k dataset from the smallest distance. The number of neighboring KNN (k) is based on the value of classification. During the classification, k was determined as 10.

3.2. Experimental Setup and Collecting Data from SDN

Experimental SDN topology for collecting normal and DDoS attack data is shown in Figure 6. In the topology, there are six (PC1-PC6) virtual machines (VMs) that have Ubuntu 18.04, 1 GB RAM, 1 CPU configuration works on VirtualBox-KVM, two VM switches, one Open vSwitch (OVS), sFlow, and InfluxDB docker images, and a Phyton-based open source OpenFlow/SDN (POX) controller. sFlow, and InfluxDB, set up in the same virtual machine, were connected to the OVS switch via a VM switch. The VM switch on the users' side was connected to the OVS switch with a bridge (br0).

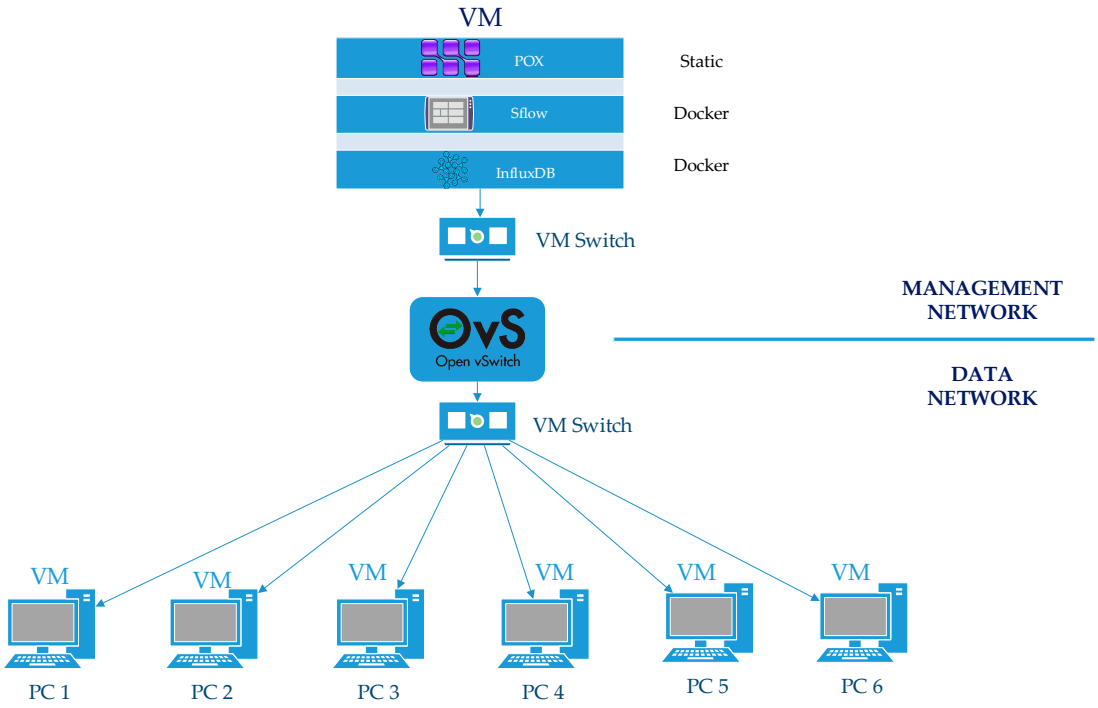

Figure 6. Experimental SDN topology for collecting data. Virtual machine (VM).

To analyze the effects of the attack traffic, network metrics were collected from the OVS switch by using an sFlowDocker image both at the moment of attack and normal traffic. For this purpose,

JavaScript was used to record network metric data on an InfluxDB image with a timestamp, which is an open source time-series platform.

We have applied a protocol-based attack scenario. Three types of flooding attacks Transmission Control Protocol (TCP), User Datagram Protocol (UDP), and Internet Control Protocol (ICMP) were used to create a DDoS attack dataset with an "hping3" packet generator. The Hping3 tool was installed on PC1 in the network and defined as attacker, while PC6 was chosen as the victim. Under attack, PC6 has the 10.0.0.6 IP address with a constant 512 bytes of payload size and constant packet rates of 2000 packet per second (pps) for each protocol-based flooding attack. This leads the number of flow modification messages from a controller to be more than 1304 per second, resulting in the number of flow table entries being 6996.

Due to the generated DDoS attack packets against the controller, the processing, bandwidth, CPU capacity, and communication capacity of the controller was overwhelmed. The capacity of the flow table entries in the switch is exhausted due to the unnecessary flows that the controller creates because of attacking packets. The simulation was conducted in 15 min for each of the sent TCP, UDP, and ICMP packets. By this means, a dataset having 65,000 samples, related to network traffic flow at the moment of DDoS attack, was generated.

In order to obtain normal traffic data for the training and testing, the TPC, UDP, and ICMP traffic data of PC2 and PC5 were recorded. By this means, a dataset, which consists of 64,000 samples where the data related to normal network traffic flow, was obtained. Thus, a new dataset, consisting of 12 features and a total of 129,000 samples, was obtained. Twelve features and the class of the dataset and their explanations are given below.

(1) Ifinpkt: Packets entering into the system from data plane.
(2) Ifoutpkt: The out packet controls the packet leaving the system according to the flow rule sent by the controller with the "Packet-out" message.
(3) Hits: The number of packets matching with the rules previously set by the controller and stored in the flow table of OVS switch.
(4) Masks: The "masks" row displays the mega flow mask stats. This row is omitted for data paths not implementing mega flow.
(5) Miss: When a new packet arrives at the transmission devices (switch, router, etc.), a match is sought in the flow table. If the arriving packet does not match with any flow input (flow miss), it is forwarded to the controller to set a new flow rule.
(6) OpenFlowfrom: The flow modification message sent by the controller. It is one of the main messages sent by the controller. With this message, switch state is changed for the new flow.
(7) Cpu_util (CPU Usage in Data Plane): The central processor unit of the computer processes and calculates the arriving data with the help of mathematical operations. The usage percentage of this unit varies according to the processing capacity of the computer. If the percentage is high, it means that the computer performs at maximum or above the normal level for the number of operating applications.
(8) Mngmnt_interface_inpkts: Packet arriving at the control plane.
(9) Mngmnt_interface_outpkts: Packet leaving the control plane.
(10) OpenFlowto: This message is one of the main messages sent by the controller. With this message, switch state is changed for the new flow.
(11) OVS Flow: The number of flows in the data path.
(12) OVS Lost: The number of packets sent for processing but dropped.
(13) Class: The class of traffic. Data used in this study are labelled data. In other words, supervised learning is aimed at four classes: 'Normal traffic', 'ICMP Flood', 'TCP Flood', and 'UDP Flood' attack.

The dataset created within the scope of the study was reduced by using feature selection methods and then the classification was made on these data. SVM, NB, ANN, and KNN were used as classifiers.

The experimental study consisted of two stages. Classifiers were trained and tested with using all the features in the dataset. Using the k-fold cross validation (k = 10) method, the dataset was divided into training and testing sets to test the success of the applied method [24] in the first stage. In the second stage, feature selection methods were used. Relief, sequential forward floating selection, and Lasso algorithms were used to select the most effective feature in the entire dataset as filter, wrapper, and embedded feature selection methods, respectively. Three different datasets were created from these feature selection methods. For each of the three datasets, the best performance ratio, between 1 and 12 features, was determined by SVM, NB, ANN, and KNN algorithms, separately.

4. Experimental Results

In this study first, data at normal traffic moments and at DDoS attack traffic moments were analyzed on SDN architecture with the dataset. Feature selection methods were not used at this stage of the study. The obtained results of the analysis are presented in Table 1.

Table 1. Performance of the machine learning models without a feature selection method.

Classifier	Accuracy	Sensitivity	Specificity	Precision	F1_Score
SVM	92.11%	88.71%	96.93%	91.42%	89.91%
KNN	95.67%	93.87%	98.01%	97.05%	95.30%
ANN	91.07%	87.27%	96.58%	89.89%	88.45%
NB	94.48%	91.77%	98.29%	92.94%	91.79%

When the success rates are analyzed, KNN and NB exhibited better performance in analyzing SDN data. At the end of the training carried out with the KNN algorithm, a Receiver Operating Characteristic (ROC) curve was prepared (Figure 7). The ROC curve is a tool for evaluating the test results. It is defined by a two-dimensional graph which has True Positive (TP) as Y-axis and False Positive (FP) as X-axis. TP indicates positive instances correctly classified as positive outputs and FP indicates negative instances wrongly classified as positive outputs. Another metric used in confusion matrices is False Negative (FN) which indicates positive instances wrongly classified as negative output. The TP value was observed to be above 0.9% in ROC curve.

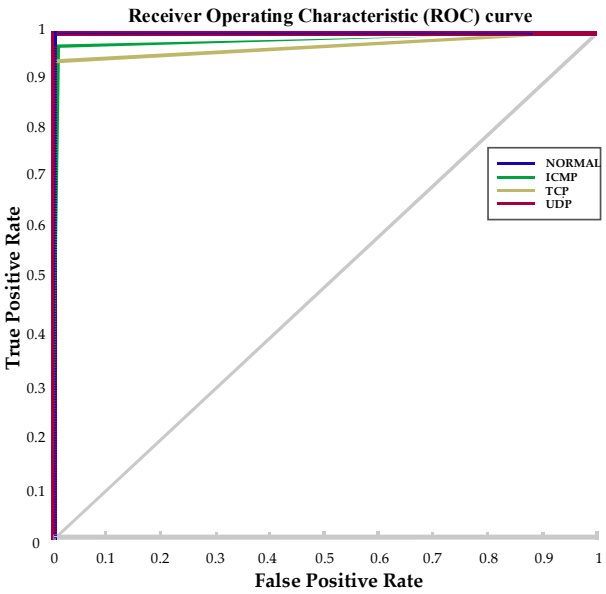

Figure 7. The normal, ICMP, TCP and UDP ROC curves for KNN without future selection.

When the performance results were examined, it was found that the KNN classifier was more successful in analyzing the attack data. The ROC curve is shown in Figure 7, and the confusion matrix

shown in Table 2 was prepared according to the results of the KNN classifier. We observed that the attack traffic type (ICMP, TCP, and UDP) affects the performance rate. The highest performance rate was obtained when the UDP flood attack was performed.

Table 2. Confusion matrix of the KNN classifier without feature selection.

		Predicted Class					
		Normal	**ICMP**	**TCP**	**UDP**	**TP**	**FN**
Actual Class	**Normal**	99.61%	0.07%	0.11%	0.18%	99.6%	0.4%
	ICMP	7.31%	90.99%	1.68%		91.0%	9.0%
	TCP	12.78%	1.15%	85.63%	0.41%	85.6%	14.4%
	UDP	0.44%	0.33%		99.21%	99.2%	0.8%

In the second part of the study, in order to obtain the features that have the most powerful effect on the prediction of machine learning models, filtering, wrapping, and embedded-based feature selection methods were applied. The reduced feature dataset obtained by the feature selection methods was applied to each model and the accuracy of the model was tested.

The Relief algorithm was applied to the SVM, KNN, ANN, and NB models as a filtering method. The proximity of each sample in this algorithm dataset to the other samples in its own class and its distance to different classes were calculated. As a result of this calculation, a model was created and feature selection was performed. As a result, 10 of the 12 features were selected for SVM, 6 for KNN, 6 for ANN, and 8 for NB. These features were retrained with each classification algorithm. The results are summarized in Table 3.

Table 3. Filter-based feature selection method performance of the machine learning models.

Classifier	**Accuracy**	**Sensitivity**	**Specificity**	**Precision**	**F1_Score**
SVM (10 Features)	92.46%	89.13%	97.02%	92.02%	90.43%
KNN (6 Features)	97.15%	95.88%	98.68%	98.10%	96.92%
ANN (6 Features)	92.28%	89.02%	96.99%	91.62%	90.20%
NB (8 Features)	95.70%	93.49%	98.65%	95.07%	93.60%

When the Relief algorithm was applied as the filtering method and the feature selection was made, the highest performance rate was obtained with the KNN classifier (Table 2). The confusion matrix of the classifier with the highest performance rate is given in Table 4.

Table 4. Confusion matrix of the KNN classifier with filter-based feature selection.

		Predicted Class					
		Normal	**ICMP**	**TCP**	**UDP**	**TP**	**FN**
Actual Class	**Normal**	99.84%	0.07%	0.07%		99.8%	0.2%
	ICMP	7.20%	91.66%	1.12%		91.7%	8.3%
	TCP	6.60%	0.94%	92.45%		92.5%	7.5%
	UDP	0.11%	0.11%	0.22%	99.55%	99.6%	0.4%

The Sequential Forward Floating Selection algorithm was applied to the SVM, KNN, ANN, and NB models with the wrapper method. A feature alone may not reduce the error, but evaluating this feature relative to another feature may improve the classification performance. For this purpose, Floating Algorithms have been developed. These algorithms have a floating variable and the number

of features added or removed at each stage is not constant. This creates a plurality of feature subset combinations [25]. Sequential Feature Selection adds or subtracts the feature based on a user-defined classifier performance metric.

In wrapper-based feature selection algorithm, when the features are reduced with the Sequential Forward Floating Selection algorithm, 8 of the 12 features were selected for SVM, 6 for KNN, 10 for ANN, and 8 for NB. The reduced dataset was rearranged for each classification algorithm. The results are shown in Table 5.

Table 5. The wrapper-based feature selection method performance of the machine learning models.

Classifier	Accuracy	Sensitivity	Specificity	Precision	F1_Score
SVM (10 Features)	92.15%	90.20%	97.26%	90.23%	90.21%
KNN (6 Features)	98.30%	97.73%	99.45%	97.72%	97.70%
ANN (6 Features)	91.44%	87.82%	97.31%	88.11%	87.89%
NB (8 Features)	94.87%	92.05%	98.43%	93.29%	92.01%

The highest performance rate was obtained with the KNN classifier when the Sequence Forward Floating Selection algorithm was applied with the wrapping method and feature selection was made (Table 5). The confusion matrix of the classifier with the highest performance rate is shown in Table 6.

Table 6. Confusion matrix of the KNN classifier with wrapper-based feature selection.

		Predicted Class					
		Normal	**ICMP**	**TCP**	**UDP**	**TP**	**FN**
Actual Class	**Normal**	99.58%	0.15%	0.15%	0.11%	99.6%	0.4%
	ICMP	0.56%	97.97%	1.12%	0.33%	98.0%	2.0%
	TCP	0.52%	5.45%	94.02%		94.0%	6.0%
	UDP	0.22%	0.11%	0.33%	99.32%	99.3%	0.7%

Finally, the SVM, KNN, ANN, and NB models were applied with the Lasso method as the embedded-based method. Lasso is another regularized variant of linear regression. An important characteristic of Lasso regression is that it eliminates the weights of the least important features. That is, in this way, it naturally makes feature selection and outputs a reduced set of features. With this method, 10 out of 12 attributes were selected for SVM, 8 for KNN, 6 for ANN, and 10 for NB. The reduced dataset was rearranged for each classification algorithm. The results are shown in Table 7.

Table 7. Embedded feature selection method performance of the machine learning models.

Classifier	Accuracy	Sensitivity	Specificity	Precision	F1_Score
SVM (10 Features)	92.15%	90.20%	97.26%	90.23%	90.21%
KNN (8 Features)	98.30%	97.73%	99.45%	97.72%	97.70%
ANN (6 Features)	91.44%	87.82%	97.31%	88.11%	87.89%
NB (10 Features)	94.87%	92.05%	98.43%	93.29%	92.01%

When the embedded method, Lasso, was applied and feature selection was made, the highest performance rate was obtained with the KNN classifier (Table 7). The confusion matrix of the classifier with the highest performance rate is shown in Table 8.

Table 8. Confusion matrix of the KNN classifier with embedded feature selection.

		Predicted Class					
		Normal	**ICMP**	**TCP**	**UDP**	**TP**	**FN**
Actual Class	**Normal**	99.39%	0.15%	0.33%	0.11%	99.4%	0.6%
	ICMP	0.78%	97.40%	1.35%	0.45%	97.4%	2.6%
	TCP	0.41%	15.30%	84.06%	0.20%	84.1%	15.9%
	UDP	0.33%	0.22%	0.44%	98.98%	99.0%	1.0%

Throughout the study, all dataset features were used and two different methods were used for reduction (Table 9). Although the results of classification using all features are acceptable, the performance results increased with the use of feature selection methods.

Table 9. Comparison of the machine learning model performances with use of feature selection methods.

Classifier	Method	No. of Features	Accuracy	Sensitivity	Specificity	Precision	F1_Score
SVM	-	12	92.11%	88.71%	96.93%	91.42%	89.91%
	Filter	10	92.46%	89.13%	97.02%	92.02%	90.43%
	Wrapper	8	92.15%	90.20%	97.26%	90.23%	90.21%
	Embedded	10	92.46%	90.73%	97.41%	90.49%	90.60%
KNN	-	12	95.67%	93.87%	98.01%	97.05%	95.30%
	Filter	6	97.15%	95.88%	98.68%	98.10%	96.92%
	Wrapper	6	98.30%	97.73%	99.45%	97.72%	97.70%
	Embedded	8	96.30%	94.95%	98.85%	95.09%	94.80%
ANN	-	12	91.07%	87.27%	96.58%	89.89%	88.45%
	Filter	6	92.28%	89.02%	96.99%	91.62%	90.20%
	Wrapper	10	91.44%	87.82%	97.31%	88.11%	87.89%
	Embedded	6	92.09%	88.91%	97.42%	89.22%	89.06%
NB	-	12	94.48%	91.77%	98.29%	92.94%	91.79%
	Filter	8	95.70%	93.49%	98.65%	95.07%	93.60%
	Wrapper	10	94.87%	92.05%	98.43%	93.29%	92.01%
	Embedded	10	95.09%	93.34%	98.45%	93.44%	93.18%

The highest accuracy rate obtained by applying the test data to the machine learning model, shown in Table 5, was obtained with the KNN classifier (98.3%) when the wrapper feature selection method was applied. This accuracy rate was obtained through training and testing on six features. We observed that the performance ratios obtained by training classifiers over a dataset containing 12 features increased after using feature selection algorithms. To compare our findings with similar studies, we created a comparison table. The results, as well as the ones reported in the literature, are summarized in Table 10.

Table 10. Comparison of the results with similar works in the literature.

Dataset Used	Feature Selection Evaluators	Machine Learning Algorithm	Accuracy (%)	Ref.
NSL-KDD DATASET (2009)	Without feature selection	SMO	97.31	[9]
		J48	98.78	
		NB	95.11	
	Genetic Alg.+ CFS	SMO	96.85	
		J48	98.33	
		NB	82.62	
	Greedy Alg. + CFS	SMO	95.71	
		J48	92.74	
		NB	83.92	
Their own dataset	Without feature selection	SVM	96.25	[10]
Their own dataset	Without feature selection	SVM	Average 95.24	[16]
LongTail	Without feature selection	C4.5 Bayesian Network DT Naive Bayes	Average 91.68	[18]
Their own dataset	Without feature selection	SOM LVQ	Average 94	[19]
CAIDA DDoS 2007	Without feature selection	SVM	92.9	[21]
Our dataset	Filter	SVM	92.46	
		KNN	97.15	
		ANN	92.28	
		NB	95.70	
	Wrapper	SVM	92.15	
		KNN	98.30	
		ANN	91.44	
		NB	94.87	
	Embedded	SVM	92.46	
		KNN	96.30	
		ANN	92.09	
		NB	95.09	

It can be seen from Table 10, that the proposed machine learning model with feature selection approaches, results in better performance in comparison with the results from [10,16,18,19,21]. It should be noted that similar studies in the literature used different datasets and different models. Therefore, justification of the comparison results is difficult.

Results show that the SDN structure proved successful in terms of detecting DDoS attacks with machine learning techniques. With the approaches to be planned on SDN architecture, a secure and efficient mechanism on the network can be developed. The performance of the controllers on large networks can be increased by catching the packets on the traffic and comparing them with the previously learned flow data of the packet. Another approach could be the use of a two-stage system in detecting DDoS attacks on the dataset. While the coming flow data is processed by the controller, it is copied to another module and analyzed by this module. To have any output on the network, the module has to inform the controller whether there is a problem in the coming data or not. Thus, while the controller performs only the network processes, the other module becomes an attack detection system.

In an SDN topology, multiple controllers in the SDN in a hierarchical structure can be located, as discussed in [26]. In SDN topology with a multi-controller structure, as the network traffic load is distributed among controllers, detection of DDoS attack traffic will be faster than that of a single controller. The location of the controllers in the network is also important at this point. Once the controllers are properly located in the network, the effectiveness of the controllers on the network

elements in the data plane will increase and delays will be decreased by optimizing resource utilization. However, in a network where multiple controllers are used, the data flow between controllers should not be interrupted for network service continuity.

In attacks such as DDoS, the attacker may target blocking the communication between controllers. SDN architecture may crash if communication between the controllers is blocked. In our study, a basic SDN topology with a single controller was used to generate DDoS attack traffic and to obtain the dataset, then to classify DDoS attacks with machine learning models, using this dataset. The use of a single-controller or multi-controller structure in the network topology at the stage of obtaining a dataset from SDN will not make a significant difference in methodology. The machine learning model that we have chosen, as the best among the models trained with the created dataset, can be applied to SDN topologies that contain both single- and multi-controllers.

5. Conclusions

In this study, SDN-based detection systems developed for DDoS attacks were analyzed with machine learning systems. In the first proposed approach, by analyzing flow data, algorithms with 98.3% accuracy ensure the detection of attacks without discriminating the type of traffic. Among the proposed systems, the second approach proceeds by labelling DDoS attacks as normal traffic and attack traffic. With 97.7% sensitivity, KNN algorithms can perform this control by lightening the burden of the controller.

With the feature selection methods used in the study, initially 12 features were selected and the selected subset of features was trained using classifiers. The number of features selected was determined either by the algorithm itself or by the threshold value given to the algorithm. By changing this threshold value, different numbers of features can be selected and trained by the classifier. Different numbers of features can change the accuracy. In general, we observed that the performance rate of all models is above 80% and the algorithms used for this dataset are successful. At the same time, network browsing, attacks between layers, and malicious software can be detected on SDN with this approach. For protecting and improving SDN structure, the second approach can be employed.

Author Contributions: Conceptualization, H.P. and A.C.; methodology, H.P. and O.P.; software, O.P.; validation, H.P., A.C. and O.P.; formal analysis, H.P. and O.P.; investigation, H.P. and A.C.; resources, O.P.; data curation, O.P.; writing—original draft preparation, H.P. and A.C.; writing—review and editing, H.P. and A.C.; visualization, A.C. and O.P.; supervision, H.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Open Networking Foundation. Software-Defined Networking: The New Norm for Networks. Available online: https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf (accessed on 31 January 2020).
2. Kreutz, D.; Ramos, F.M.; Verissimo, P.E.; Rothenberg, C.E.; Azodolmolky, S.; Uhlig, S. Software-defined networking: A comprehensive survey. *Proc. IEEE* **2015**, *103*, 14–76. [CrossRef]
3. Xie, J.; Guo, D.; Hu, Z.; Qu, T.; Lv, P. Control plane of software defined networks: A survey. *Comput. Commun.* **2015**, *67*, 1–10. [CrossRef]
4. Suciu, G.; Vulpe, A. An Overview Study of Software Defined Networking. In Proceedings of the IE International Conference, Bucharest, Romania, 30 April–3 May 2015.
5. Kreutz, D.; Ramos, F.; Verissimo, P. Towards secure and dependable software-defined networks. In Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, New York, NY, USA, 16 August 2013.
6. Shu, Z.; Wan, J.; Li, D.; Lin, J.; Vasilakos, A.V.; Imran, M. Security in software-defined networking: Threats and countermeasures. *Mob. Netw. Appl.* **2016**, *21*, 764–776. [CrossRef]

7. Ali, S.T.; Sivaraman, V.; Radford, A.; Jha, S. A survey of securing networks using software defined networking. *IEEE Trans. Reliab.* **2015**, *64*, 1086–1097. [CrossRef]
8. Kuerban, M.; Tian, Y.; Yang, Q.; Jia, Y.; Huebert, B.; Poss, D. FlowSec: DOS Attack Mitigation Strategy on SDN Controller. In Proceedings of the IEEE International Conference on Networking, Architecture and Storage (NAS), Long Beach, CA, USA, 8–10 August 2016.
9. Alshamrani, A.; Chowdhary, A.; Pisharody, S.; Lu, D.; Huang, D. A defense system for defeating DDoS attacks in sdn based networks. In Proceedings of the 15th ACM International Symposium on Mobility Management and Wireless Access, Miami Beach, FL, USA, 21–25 November 2017.
10. Latah, M.; Toker, L. A novel intelligent approach for detecting DoS flooding attacks in software-defined networks. *Int. J. Adv. Intell. Inform.* **2018**, *4*, 11–20. [CrossRef]
11. Li, C.; Wu, Y.; Yuan, X.; Sun, Z.; Wang, W.; Li, X.; Gong, L. Detection and defense of DDoS attack–based on deep learning in OpenFlow-based SDN. *Int. J. Commun. Syst.* **2018**, *31*, e3497. [CrossRef]
12. Li, W.; Meng, W.; Kwok, L.F. A survey on OpenFlow-based Software Defined Networks: Security challenges and countermeasures. *J. Netw. Comput. Appl.* **2016**, *68*, 126–139. [CrossRef]
13. Padmaja, S.; Vetriselvi, V. Mitigation of switch-Dos in software defined network. In Proceedings of the International Conference on Information Communication and Embedded Systems (ICICES), Chennai, India, 25–26 February 2016.
14. Polat, H.; Polat, O. The effect of DoS attack on ODL and POX SDN controllers. In Proceedings of the 8th International Conference on Information Technology (ICIT), Amman, Jordan, 17–18 May 2017.
15. Bholebawa, I.Z.; Dalal, U.D. Design and Performance Analysis of OpenFlow-Enabled Network Topologies Using Mininet. *Int. J. Comput. Commun. Eng.* **2016**, *5*, 419–429. [CrossRef]
16. Ye, J.; Cheng, X.; Zhu, J.; Feng, L.; Song, L. A DDoS attack detection method based on SVM in software defined network. *Secur. Commun. Netw.* **2018**, *2018*, 1–8. [CrossRef]
17. Xue, L.; Yuan, D.; Hu, H.; Ran, J.; Li, S. DDoS detection in SDN switches using support vector machine classifier. In Proceedings of the Joint International Mechanical, Electronic and Information Technology Conference (JIMET-15), Chongqing, China, 18–20 December 2015.
18. Nanda, S.; Zafari, F.; DeCusatis, C.; Wedaa, E.; Yang, B. Predicting network attack patterns in SDN using machine learning approach. In Proceedings of the 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), Palo Alto, CA, USA, 7–10 November 2016.
19. Jankowski, D.; Amanowicz, M. On efficiency of selected machine learning algorithms for intrusion detection in software defined networks. *Int. J. Electron. Telecommun.* **2016**, *62*, 247–252. [CrossRef]
20. Chakraborty, S.; Banerjee, S. Proposed approach to detect distributed denial of service attacks in software defined network using machine learning algorithms. *Int. J. Eng. Technol.* **2018**, *7*, 472–476.
21. Mowla, N.I.; Doh, I.; Chae, K. CSDSM: Cognitive switch-based DDoS sensing and mitigation in SDN-driven CDNi word. *Comput. Sci. Inf. Syst.* **2018**, *15*, 163–185. [CrossRef]
22. Hira, Z.M.; Duncan, F.G. A Review of Feature Selection and Feature Extraction Methods Applied on Microarray Data. *Adv. Bioinform.* **2015**, *2015*, 1–13. [CrossRef] [PubMed]
23. Venkatesh, B.; Anuradha, J. A Review of Feature Selection and Its Methods. *Cybern. Inf. Technol.* **2019**, *19*, 3–26. [CrossRef]
24. Stamp, M. *Introduction to Machine Learning with Applications in Information Security*; Chapman and Hall/CRC: Boca Raton, FL, USA, 2017.
25. Somol, P.; Novovicova, J.; Pudil, P. Improving Sequential Feature Selection Methods' Performance by Means of Hybridization. In Proceedings of the IASTED International Conference Advances in Computer Science and Engineering, Sharm El Sheikh, Egypt, 15–17 March 2010.
26. Mostafaei, H.; Menth, M.; Obaidat, M.S. A Learning Automaton-Based Controller Placement Algorithm for Software-Defined Networks. In Proceedings of the 2018 IEEE Global Communications Conference (GLOBECOM), Abu Dhabi, UAE, 9–13 December 2018.