DDoS Attack Detection and Mitigation in SDN using Machine Learning

Fatima Khashab[1,2], Joanna Moubarak[1,3], Antoine Feghali[3], and Carole Bassil[2]

[1]University of Saint Joseph, Beirut, Lebanon
fatima.khashab@net.usj.edu.lb, joanna.moubarak1@usj.edu.lb
[2]Lebanese University, Beirut, Lebanon
cbassil@ul.edu.lb
[3]Potech Labs, Beirut, Lebanon
Joanna.b@potech-consulting.com, Tony@potech.global

2021 IEEE 7th International Conference on Network Softwarization (NetSoft) | 978-1-6654-0522-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/NetSoft51509.2021.9492558

***Abstract*—Software Defined Networking (SDN) is a networking paradigm that has been very popular due to its advantages over traditional networks with regard to scalability, flexibility, and its ability to solve many security issues. Nevertheless, SDN networks are exposed to new security threats and attacks, especially Distributed Denial of Service (DDoS) attacks. For this aim, we have proposed a model able to detect and mitigate attacks automatically in SDN networks using Machine Learning (ML). Different than other approaches found in literature which use the native flow features only for attack detection, our model extends the native features. The extended flow features are the average flow packet size, the number of flows to the same host as the current flow in the last 5 seconds, and the number of flows to the same host and port as the current flow in the last 5 seconds. Six ML algorithms were evaluated, namely Logistic Regression (LR), Naïve Bayes (NB), K-Nearest Neighbor (KNN), Support Vector Machine (SVM), Decision Tree (DT), and Random Forest (RF). The experiments showed that RF is the best performing ML algorithm. Also, results showed that our model is able to detect attacks accurately and quickly, with a low probability of dropping normal traffic.**

Keywords—Software Defined Networking, Attack Detection, Attack Mitigation, Machine Learning, Feature Extending.

I. INTRODUCTION

Achieving network programmability has become an essential aim of network research due to the ever-evolving network requirements. Software-Defined Networking (SDN) is an emerging architecture aimed to address this need and has gained a lot of attention in recent years as an alternative to traditional networks. SDN separates the control plane from the data plane, making network devices simple packet forwarding elements, programmed by a logically centralized unit called the SDN controller, through a standardized protocol such as OpenFlow [1][2]. This separation offers greater control capabilities for network management, which in turn simplifies the construction of intelligent and automated networks [3]. Also, this separation offers more flexibility as it facilitates the implementation of new network applications and services [2].

Despite their great adoption, SDN-based networks are still vulnerable to many types of network intrusions and are exposed to new security threats and attacks [4], especially Distributed Denial of Service (DDoS) attacks, which can affect the resiliency and reliability of SDN networks. Anomaly traffic detection has been one of the largest challenges and major issues in the SDN research field today. The design of attack detection and mitigation techniques is facilitated in the SDN paradigm, as SDN provides a global view of the network, allows collecting flow statistics easily, and supports dynamic updating of flow entries [5]. This is much more difficult in traditional networks, where devices require exchanging lots of information in order to infer partially the state of the remaining parts of the network, leading to limited visibility and thus limited detection capabilities.

Many techniques have been proposed in literature for the detection of attacks in SDN networks [6]. Recently, Machine Learning (ML) approach is playing an essential role as it has shown promising results and has proven to perform well in detecting anomalies [7]-[9]. However, methods using ML have shown some limitations. Some of them use packet-based detection which induces much processing and memory overhead [10]. Other methods use aggregated flow features, but these are not able to indicate the source of the attack, and thus are not able to mitigate them [11][12]. Other methods have resolved the above limitations by inspecting each flow and extracting its features. However, a limitation arises in the latter framework, due to the fact that the native flow features extracted are mostly not sufficient to accurately distinguish between normal and attack flows [3]. Hence, in this paper we will extend these features in order to improve the detection performance.

In this paper, we propose an SDN model able to detect and mitigate attacks automatically in SDN networks using ML. The model collects traffic flow entries from all the switches periodically, extracts the native flow features, and then extends them. The extended flow features are the average flow packet size, the number of flows to the same host as the current flow in the last 5 seconds, and the number of flows to the same host and port as the current flow in the last 5 seconds. These extended features in addition to the native counters (i.e. packet count and byte count) are then used by a detection module to classify each flow as either normal or anomaly using a binary classification ML algorithm. Once an attack is detected, the source of the attack is blocked. Concerning the classification ML algorithm used in the detection module, six ML algorithms were evaluated, namely Support Vector Machine (SVM) [13], Logistic Regression (LR) [14], K-Nearest Neighbor (KNN) [15], Decision Tree (DT) [16], Naïve Bayes (NB) [17], and Random Forest (RF) [18].

We have developed a prototype system as a proof-of-concept. The prototype was implemented as a Python script on the top of a Floodlight controller [19] using REST API. To evaluate our proposed method, we simulated an SDN network topology using Mininet emulator [20], and we used Python scripts to generate normal and attack traffic. Also,

978-1-6654-0522-5/21/$31.00 ©2021 IEEE

we compared our approach with the basic approach which doesn't consider extending features.

The rest of this paper is organized as follows. Section II reviews some related work. Section III describes the proposed framework for attack detection and mitigation in SDN networks. Section IV presents the evaluation of the framework, including the evaluation of the different ML algorithms, a comparison of our model with the basic model, and the evaluation of the overall detection and mitigation model. Section V concludes this paper presenting final remarks and future work.

II. Related Work

Most of the work previously done have focused on anomaly detection. Few are the researches that have coped with anomaly detection and mitigation jointly. Also, most researches have focused on detecting and mitigating Denial of Service (DoS) or DDoS attacks exclusively, as these are considered among the most dangerous attacks in SDN networks. Recently, researchers are focusing on ML algorithms for attack detection [7]-[9]. A state of the art on attack detection and mitigation in SDN networks using ML is discussed below and summarized in TABLE I.

Li et al. in [21] established a novel mechanism to protect SDN networks based on ML framework. Firstly, the scheme collects traffic data from the packet-in messages, and then extracts some key features, such as source IP address, source port, destination IP address, destination port, and protocol type. Entropy is then used to measure the distribution of each feature. The resultant features' entropy is used to train a nonlinear SVM algorithm, as it proved to perform better than other ML algorithms, including DT, NB, KNN, and RF.

Dennis et al. in [3] designed a flow-based DDoS attack detection system using RF ML algorithm with weighted voting. They used UCLA dataset and modified it by adding traffic flow entries of simulated traffic. Then, they selected the native OpenFlow counters as features, which are the number of packets, number of bytes, and flow duration, to build their model. The approach sends the flow statistics collected from the switches to the RF classifier every 10 seconds. Once an attack is confirmed, the mitigation module is called, and the attack traffic is dropped to prevent the breakdown of the switches in the network.

Ye et al. in [11] proposed a DDoS detection scheme based on SVM. The scheme collects traffic data from the flow tables periodically, and then extracts six aggregated features related to DDoS attacks, which are the speed of source IP, speed of source port, speed of flow entries, standard deviation of flow packets, standard deviation of flow bytes, and ratio of pair-flow, to use as characteristic values for SVM classification. The validity of their method was verified by simulation.

Oo et al. in [12] performed similar work as in [4] for the detection of DDoS attacks in SDN networks. They proposed Advanced SVM (ASVM) algorithm as an enhancement of the existing SVM. Traffic data is collected from the flow tables periodically, and then five aggregated traffic features are extracted, namely the average number of flow packets, average number of flow bytes, variation of flow packets, variation of flow bytes, and the average duration of traffic in the sampling interval. In order to train and test their model, they simulated an SDN network and generated normal and attack traffic.

Rahman et al. in [22] designed an SDN framework to detect and mitigate DDoS attacks (ICMP and TCP floods) using ML. They synthesized a dataset containing normal and DDoS traffic of 24 packet-level data features, and then used it to select the best performing classifier among four machine learning techniques, which are J48, RF, SVM, and K-NN. A detection script was built using the best performing classifier, J48, to classify incoming packets as normal or attack, and a mitigation script was used to block the attackers' ports on the switch for 30 seconds in case a DDOS attack was detected.

Alshamrani et al. in [23] developed a secure system to detect and mitigate DDoS attacks using ML. Their detection module includes two algorithms for offline and online DDoS detection. First, they applied feature selection algorithm on NSL-KDD dataset. Then, they employed SVM to offline train and test their system after comparing with J48 and NB. To online evaluate their trained model, they deployed it as a REpresentational State Transfer Application Programming Interface (REST API) on the top of the SDN controller. Their system is based on collecting packets periodically and extracting 24 features from each packet. Upon attack detection, malicious traffic is forwarded to a honeypot responsible for deeply learning and analyzing the attackers' behaviors.

However, the above-mentioned approaches have some drawbacks. In [11], [12], and [21], although the use of entropy or aggregated features causes remarkable overhead reduction to the detection mechanism, these approaches are not able to indicate the source of the attack, and thus are not able to mitigate them. This has been resolved in [3] by inspecting each flow and extracting its features. However, a limitation arise in this framework, which is due to the fact that the native flow features extracted are mostly not sufficient to accurately distinguish between normal and attack flows. Also, the use of the duration feature in [3] is not suitable, since the duration in the flow entries does not represent the duration of the flow, but rather represent the duration since the flow entry was installed in the flow table. The drawback in [22] and [23] is that the use of packet-based detection induces much processing and memory overhead. The limitations of the above approaches are resolved in this paper, where flow-based detection using ML is adopted while extending the flow features.

TABLE I. Summary of Related Works for ML-based Defence Solutions in SDN

Reference	Scope	ML Algorithms	Features	Dataset	Limitation
[21]	Detection	SVM	5 entropy features	Synthetic	Not able to indicate source of attack
[3]	Detection Mitigation	RF	3 native features	UCLA dataset + Synthetic	Not accurate

[11]	Detection	SVM	6 aggregated features	Synthetic	Not able to indicate source of attack
[12]	Detection	Advanced SVM	5 aggregated features	Synthetic	not able to indicate source of attack
[22]	Detection Mitigation	J48 RF SVM K-NN	24 packet-level features	Synthetic	overhead
[23]	Detection Mitigation	SVM J48 NB	25 features	NSL-KDD [24]	overhead

III. SDN Detection and Mitigation Model

In order to achieve our goal of building attack detection and mitigation model for SDN networks, we developed a model on the application plane. The architecture of our proposed mechanism consists of four main modules: the Flow Collector module, the Feature Extender module, the Anomaly Detection module, and the Anomaly Mitigation module. The Flow Collector is responsible for collecting traffic flow entries from all the flow tables of all the switches periodically. These flow entries will be transferred to the Feature Extender that will derive new features for each flow entry. Then, the information generated will be delivered to the Anomaly Detection module which performs flow-based detection using ML, and classifies each flow as normal or anomaly. If a flow is classified as anomalous, it will be transferred to the Mitigation module in order to block the source of the attack; else no action will be taken. The basic components of our framework are depicted in Fig. 1. Next, we describe these components in details.

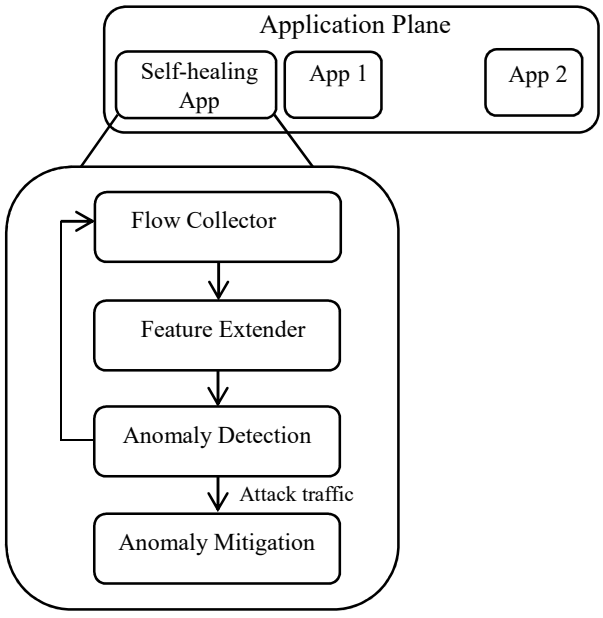

Fig. 1. Basic components of our self-healing framework.

A. Flow Collector Module

Initially, the Flow Collector communicates with the controller to request traffic information. Then, triggered by the traffic information request, the controller uses the OpenFlow protocol to collect flow tables' information. The controller sends a *flow-stats request*, a message that queries the switch for flow statistics, to every switch associated with the controller. Consequently, each switch replies with a *flow-stats reply* message containing all flow entries in all flow tables, with each entry containing the description of the flow along with its corresponding counters. The controller then gathers the traffic information from all the switches and replies to this component.

After receiving the flow information, the Flow Collector parses this information and removes irrelevant data. The relevant data retained are the flow identifier (flow-id) and the flow counters. We define the flow-id as a seven-tuple consisting of source IP address (srcip), destination IP address (dstip), source MAC address (srcMAC), destination MAC address (dstMAC), TCP/UDP source port (srcport), TCP/UDP destination port (dstport), and transport protocol identification (protocol). It is important to note that this tuple can be adjusted according to the needs of the network infrastructure. The flow counters provided by OpenFlow consist of only three counters, which are the packet count, byte count, and duration.

Following this, this component organizes relevant data using a dictionary data structure indexed by the flow-id, with each flow entry structured/summarized as follows:

<srcip, dstip, srcport, dstport, protocol> : <pkt_count, byte_count, duration >

The collection of flow entries from OpenFlow switches was performed at predetermined periodic time intervals by the controller. The definition of this time interval is of great importance. If the collection is made at infrequent time intervals, then there will be a delay to detect an attack and consequently a reduction of the time available for a possible mitigation. On the other hand, if the time interval for collection is too short, this will lead to an increase in the overhead of the detection mechanism. In general, this interval is estimative and depends on several environmental aspects, such as communication delay, network topology, and traffic. In our case, the polling interval was set to 5 seconds in order to obtain a fine-grained view of the network traffic.

B. Feature Extender Module

The Feature Extender uses the native counters obtained for the current time interval to calculate new flow features. The main reason for calculating extended flow features is that OpenFlow offers only a limited number of flow features to describe the traffic profile. Although traffic classification can be performed using the native counters, these extended flow features can improve the performance of classification schemes. Also, classification using native counters does not allow detecting some anomaly traffic profiles, so more descriptive traffic discriminators are necessary.

Therefore, three new flow features are derived, namely the average flow packet size (pkt_size), the number of flows to the same host as the current flow in the last 5 seconds (same_host), and the number of flows to the same host and port as the current flow in the last 5 seconds (same_host_port). Since calculating the exact value of the packet size would require deep packet inspection and monitoring of every packet in the network, estimation (i.e. average) of this value is calculated as shown in equation (1).

Concerning same_host and same_host_port features, they are derived by inspecting and comparing the collected flows.

$$pkt\_size = \frac{Byte\ Count}{Packet\ Count} \quad (1)$$

C. Anomaly Detection Module

Intrusion detection could be categorized in many ways, whether depending on the source of data to be analyzed or regarding the technique used to detect unusual events. Based on data to be analyzed, flow-based or packet-based detection is used, and based on the detection technique signature-based or anomaly-based detection can be used. Based on the source of data to be analyzed, flow-based detection was chosen, as this would be more efficient than packet-based detection in terms of processing and memory overhead and thus is more suitable for high-speed networks. As a detection technique, we chose anomaly-based intrusion detection, and particularly detection using ML. The reason behind our choice is that the malicious traffic can be of any form, and the attackers can change their attack patterns regularly. Hence, there is a need to learn from experience, and which can be achieved by the aid of ML.

We can think of attack detection as a binary classification problem that classifies each flow as either normal or anomaly. In our approach, we used offline ML, where it is essential first to build a model using a classification ML algorithm, and then use this model to classify network flows. This ML process is illustrated in Fig. 2. Note that different classification ML algorithms can be used to construct the detection model. In our implementation, we used the most appropriate algorithm among 6 ML algorithms, which are LR, NB, KNN, SVM, DT, and RF, based on some evaluation metrics.

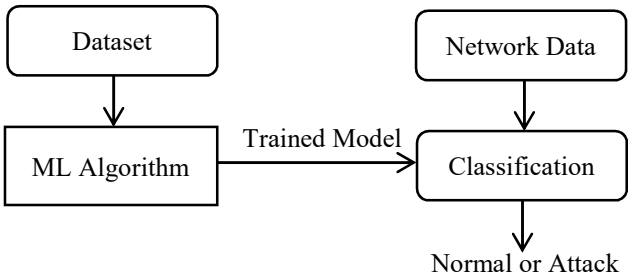

Fig. 2. ML process in an SDN environment.

The choice of classification learning imposes the need to use a labeled dataset to train the ML model, so a flow-based labeled dataset that represents network traffic is needed. Moreover, the choice of features to train the model is very crucial as it influences ML performance to a great extent. We chose 5 features to build our model, which are pkt_count, byte_count, pkt_size, same_host, and same_host_port.

After building the ML model, flows from the Feature Extender module will arrive to the Detection module every 5 seconds. This module will use the above mentioned 5 features to indicate if an anomaly has been identified for a specific flow-id using the built model. Flows that are considered normal do not require any action. However, malicious flows are subsequently forwarded to the Anomaly Mitigation module.

D. Anomaly Mitigation Module

The Anomaly Mitigation module is responsible for taking mitigation actions once a malicious flow is confirmed by the anomaly detection module, in order to avoid network disruption or performance degradation. In our framework, we aim to block the source of the detected attack. Note that blocking the IP addresses of the attackers does not always help in mitigating attacks due to IP spoofing. In our scenario, we block the attacker's Ethernet address for proof-of-concept.

IV. Simulation and Results

After designing our proposed attack detection and mitigation method, it is now essential to test and evaluate it. For this aim, we have developed a prototype system as a proof-of-concept. The evaluation of the framework includes the evaluation of the different ML algorithms, a comparison of our model with the basic model, and the evaluation of the overall self-healing model.

A. Simulation Profile

Building a dataset is the most important step in the implementation of classification ML algorithms. In our case, we generated a dataset by simulation because we did not find a public dataset suitable for our approach. We have researched numerous publicly available datasets [25][26]; however they have shown several limitations. Some were packet-based datasets; others did not contain normal traffic data, and some were unlabeled. The other datasets that did not contain the previously mentioned limitations contained a limited number of features. Therefore, we had to simulate our own dataset.

For evaluating our proposed method, we first established an SDN topology using Mininet [20] emulator. To achieve this, we used two Ubuntu virtual machines on VMWare Workstation, one for the Floodlight controller, and the other for Mininet. We built a tree-type network of depth two with one controller, four OpenFlow v1.3 switches, and nine hosts as shown in Fig. 3.

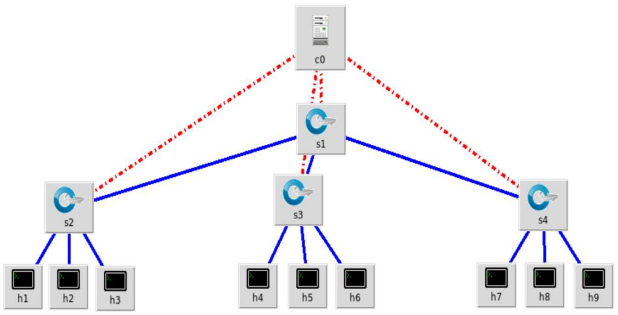

Fig. 3. Simulation network topology.

After the SDN network topology was set up, we used two Scapy [27] programs to generate normal and attack traffic. We ran the normal traffic from randomly chosen 5 hosts to three different destinations, and we generated the attack traffic from the remaining host toward one of these destinations. TABLE II describes the normal traffic used in our experiments. The attack traffic we simulated is a TCP flood attack using spoofed IP address.

TABLE II. NORMAL TRAFFIC PATTERN

Parameter	Value
Packet Type	TCP
Packet Payload	Random between 60 and 800
Number of Packets Sent per Flow	Random between 1 and 5
Packet Inter-Arrival Interval	0.1 seconds

TABLE III presents the number of generated flows for normal and attack traffic launched for the training and testing phases. The numbers in brackets are the sizes of the packets at the time of attack. We used a larger testing dataset than the training dataset with new attacks in order to ensure a more accurate calculation of the model performance.

TABLE III. TRAFFIC USED FOR TRAINING AND TESTING

Traffic Type	Training	Testing
Normal	6472	10828
TCP(200)	976	1289
TCP(500)	960	970
TCP(600)	-	1023
TCP(800)	-	1101

The training and testing samples are captured separately in 2 different csv files by the use of a script. This script collects, extends, and labels the traffic flows. Note that for simplifying the labelling, we considered that the normal traffic is generated from a different TCP source port than that of the attack. Then, the samples of each of the 2 files are randomly shuffled to create the raw data.

B. Intrusion Detection Performance Evaluation

After creating the training and testing datasets, these datasets were used to measure the performance of different ML algorithms. Various metrics can be used to measure the performance of a ML algorithm. These metrics are derived from the confusion matrix parameters.

In binary classification problems, four parameters are used to represent the confusion matrix, which are the number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). A brief explanation of these parameters is discussed below.

TP: indicates the number of anomaly flows that are correctly classified as anomaly.

FN: indicates the number of anomaly flows that are incorrectly classified as normal.

FP: indicates the number of normal flows that are incorrectly classified as anomaly.

TN: indicates the number of normal flows that are correctly classified as normal.

For our anomaly detection problem, accuracy (ACC), precision (P), recall (R), specificity (S), training time, and testing time were used as evaluation metrics. The discussions of these metrics along with their formulas are given below:

Accuracy: it is the percentage of the correctly classified flows made to both normal and anomaly classes to the total classified flows.

$$ACC = \frac{TP+TN}{TP+TN+FP+FN} \times 100\% \quad (2)$$

Precision: it is the percentage of anomaly flows that were correctly classified among all the flows classified as anomaly.

$$P = \frac{TP}{TP+FP} \times 100\% \quad (3)$$

Recall: also called sensitivity, is the percentage of anomaly flows that were correctly classified among all the actual anomalous flows.

$$R = \frac{TP}{TP+FN} \times 100\% \quad (4)$$

Specificity: is the percentage of normal flows that were correctly classified among all the actual normal flows.

$$S = \frac{TN}{TN+FP} \times 100\% \quad (5)$$

Training time: it is the time the classifier consumes to train the model.

Testing time: it is the time the classifier consumes to test the model.

Table IV illustrates the different performance metrics calculated for the different ML algorithms. Concerning accuracy, precision, recall, and specificity, we find that RF achieves the highest values of these metrics. Also, inspecting the training and testing times, we find that all the algorithms have low values which are less than 1 second. The training time is not important in our case since the model is built before online classification; however we seek low testing time in order to decrease the online classification processing overhead. Although RF doesn't have the lowest testing time, its value is still very efficient to be used. Hence, we used RF to build the training model for our detection module, as it performed better than the other algorithms.

TABLE IV. PERFORMANCE OF DIFFERENT ML MODELS

ML Algorithm	Accuracy	Precision	Recall	Specificity	Training Time (sec)	Testing Time (sec)
SVM	94.99%	97.1%	82.81%	99.95%	0.48	0.61
LR	98.9%	99.95%	96.32%	99.98%	0.216	0.0039
KNN	86.41%	99.82%	53%	99.96%	0.01	0.913
DT	99.11%	98.01%	99.01%	99.18%	0.0094	0.001
NB	99.64%	99.97%	98.98%	99.98%	0.005	0.0027
RF	99.76%	99.97%	99.29%	99.99%	0.6	0.15

In order to compare these results with the basic approach which considers only the native flow features, i.e. packet count, byte count, and duration, we used the same datasets to train and test the above ML algorithms, but by using these 3 native features only. Note that the duration represents the duration since the flow entry was installed in the flow table and not the duration of the flow. Table V presents the different performance metrics calculated for different ML algorithms when using the basic flow features.

TABLE V. PERFORMANCE OF DIFFERENT ML MODELS IN BASIC APPROACH

ML Algorithm	Accuracy	Precision	Recall	Specificity
SVM	71.17%	-	0%	100%
LR	73.25%	55.84%	34.44%	88.98%
KNN	86.53%	98.3%	54.27%	99.62%

DT	86.74%	99.62%	54.27%	99.91%
NB	76.39%	54.98%	100%	66.87%
RF	86.72%	99.45%	54.27%	99.87%

Analyzing TABLE V, the precision in SVM could not be calculated since all the flows (normal and attack flows) were classified as normal, and no attack was detected. Looking at the recall and precision for the other ML algorithms, we find that either both values are low or one is high and the other is low. Low precision and high recall implies that most attacks are detected, but many normal flows are classified as malicious. High precision and low recall signifies that normal flows are correctly classified while many attacks are not detected. Thus, we can infer that all the ML algorithms are not able to distinguish well between normal and attack traffic. Hence, it is clear that our modified approach has highly improved the detection of malicious attacks.

C. Evaluation of SDN Detection and Mitigation Model

In order to evaluate the performance of our model, we need to test the process of online traffic classification and attack mitigation. For mitigating attacks, we used the firewall REST API to create firewall rules. Thus, for every flow detected as malicious, a firewall rule is installed to block the Ethernet address from which the attack is launched.

We ran the Python code of our prototype using REST API, and then we generated normal traffic as background traffic, and we attacked from period 20 to period 30. During the simulation of traffic, we monitored the overall number of traffic flows and the number of blocked flows as illustrated in Fig. 4 and Fig. 5 respectively.

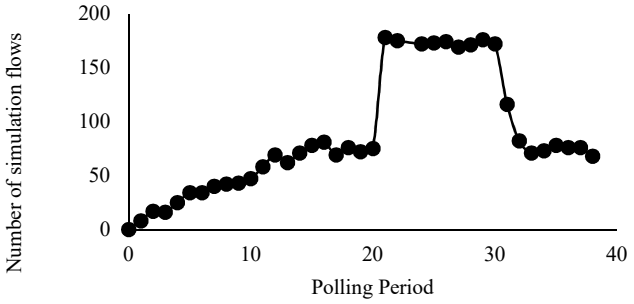

Fig. 4. The overall number of traffic flows during simulation.

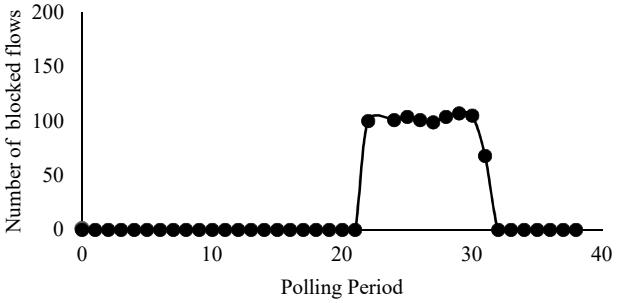

Fig. 5. Number of blocked flows during simulation.

From these two figures, we can find that our model didn't block any flow during normal traffic generation. An attack was launched at period 20, and as shown in Fig. 5, flows were blocked starting from period 22. The analysis for this is that after the attack traffic was launched at period 20, it was collected and detected at period 21 and firewall rules were installed to block the source of the attack. After this, all the attacks launched from the malicious host were blocked, and the number of blocked flows was then calculated at period 22. Thus, we can conclude that our framework was able to detect attacks quickly, mitigate them, without affecting the normal traffic.

V. Conclusion and Future Work

This paper proposed a model able to detect and mitigate DDoS attacks automatically in SDN networks using ML. The model collects traffic flow entries from all the switches periodically, extracts the native flow features, and then extends them by adding new features. Five features are then used by a detection module to classify each flow as either normal or anomaly. Once an attack is detected, the source of the attack is blocked. Concerning the classification ML algorithm used in the detection module, six ML algorithms were evaluated, namely LR, NB, KNN, SVM, DT, and RF. Our experiment results showed that RF is the most suitable classifier for our network. Our model was able to detect and block attacks accurately and quickly, without dropping normal traffic.

As a part of our future work, we aim to train and test our algorithms by using real network traffic dataset with more attacks. Furthermore, we aim to investigate new classification schemes and compare them with our selected ML algorithms. Also, we intend to use new mitigation strategies. Finally, we will try to find additional flow features in order to increase the system's performance and accuracy.

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69–74, Mar. 2008.

[2] M. Mousa, A. Bahaa-Eldin and M. Sobh, "Software Defined Networking concepts and challenges", in *2016 11th International Conference on Computer Engineering & Systems (ICCES)*, 2016, pp. 79-90.

[3] Dennis, J. Rukshan, X. Li. "Machine-Learning and Statistical Methods for DDoS Attack Detection and Defense System in Software Defined Networks", 2018.

[4] M. Dabbagh, B. Hamdaoui, M. Guizani and A. Rayes, "Software-defined networking security: pros and cons," in IEEE Communications Magazine, vol. 53, no. 6, pp. 73-79, 2015.

[5] A. Santos da Silva, J. A. Wickboldt, L. Z. Granville and A. Schaeffer-Filho, "ATLANTIC: A framework for anomaly traffic detection, classification, and mitigation in SDN," NOMS 2016 - 2016 IEEE/IFIP Network Operations and Management Symposium, Istanbul, 2016, pp. 27-35.

[6] A. Bahlali, "Anomaly-Based Network Intrusion Detection System: A Machine Learning Approach", Biskra University, 2019.

[7] J. Moubarak and T. Feghali, "Comparing Machine Learning Techniques for Malware Detection", in *ICISSP*, 2020.

[8] M. S. Elsayed, N. -A. Le-Khac, S. Dev and A. D. Jurcut, "Machine-Learning Techniques for Detecting Attacks in SDN," 2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT), Dalian, China, 2019, pp. 277-281.

[9] J. Xie et al., "A Survey of Machine Learning Techniques Applied to Software Defined Networking (SDN): Research

Issues and Challenges," in IEEE Communications Surveys & Tutorials, vol. 21, no. 1, pp. 393-430, Firstquarter 2019.

[10] H. Alaidaros, M. Mahmuddin and A. Al Mazari, "An Overview of Flow-based and Packet-based Intrusion Detection Performance in High Speed Networks," Proceedings of the International Arab Conference on Information Technology, pp. 1-9, 2011.

[11] J. Ye, X. Cheng, J. Zhu, L. Feng and L. Song, "A DDoS Attack Detection Method Based on SVM in Software Defined Network", *Security and Communication Networks*, vol. 2018, pp. 1-8, 2018.

[12] M. Myint Oo, S. Kamolphiwong, T. Kamolphiwong and S. Vasupongayya, "Advanced Support Vector Machine- (ASVM-) Based Detection for Distributed Denial of Service (DDoS) Attack on Software Defined Networking (SDN)", *Journal of Computer Networks and Communications*, vol. 2019, pp. 1-12, 2019.

[13] Y. Tian, Y. Shi, and X. Liu, "Recent advances on support vector machines research," Technological and Economic Development of Economy, vol. 18, no. 1, pp. 5–33, 2012.

[14] J. Stoltzfus, "Logistic Regression: A Brief Primer", *Academic Emergency Medicine*, vol. 18, no. 10, pp. 1099-1104, 2011.

[15] Su, Ming-Yang. "Real-time anomaly detection systems for Denial-of-Service attacks by weighted k-nearest-neighbor classifiers." *Expert Systems with Applications* 38, no. 4 (2011): 3492-3498.

[16] H. Patel and P. Prajapati, "Study and Analysis of Decision Tree Based Classification Algorithms", *International Journal of Computer Sciences and Engineering*, vol. 6, no. 10, pp. 74-78, 2018.

[17] I. Rish et al., "An empirical study of the naive bayes classifier," in Proc. IJCAI workshop on empirical methods in artificial intelligence, 2001, vol. 3, pp. 41–46.

[18] E. Goel, Er. Abhilasha, E. Goel, and E Abhilasha, "Random forest: A review," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 7, no. 1, 2017.

[19] "Project Floodlight", *Floodlight.atlassian.net*. [Online]. Available: https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/overview.

[20] M. Team, "Mininet: An Instant Virtual Network on your Laptop (or other PC) - Mininet", *Mininet.org*, 2020. [Online]. Available: http://mininet.org/.

[21] D. Li, C. Yu, Q. Zhou, J. Yu, "Using SVM to detect DDoS attack in SDN network", IOP Conference Series Material Science and Engineering, vol. 466, pp. 012003, 2018.

[22] O. Rahman, M. A. G. Quraishi and C. Lung, "DDoS Attacks Detection and Mitigation in SDN Using Machine Learning," 2019 IEEE World Congress on Services (SERVICES), Milan, Italy, 2019, pp. 184-189.

[23] A. Alshamrani, A. Chowdhary, S. Pisharody, D. Lu, and D. Huang, "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. ACM, 2017, pp. 83–92.

[24] L. Dhanabal and S. Shantharajah, "A Study on NSL-KDD Dataset for Intrusion Detection System Based on Classification Algorithms", in *International Journal of Advanced Research in Computer and Communication Engineering*, 2015.

[25] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Towards generating real-life datasets for network intrusion detection." International Journal of Network Security, vol. 17, no. 6, pp. 683–701, 2015.

[26] 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, 2019. Available: 10.1016/j.cose.2019.06.005.

[27] P. community., "Scapy", *Scapy.net*, 2020. [Online]. Available: https://scapy.net/.