Received: 16 November 2018 | Revised: 19 April 2019 | Accepted: 18 May 2019
DOI: 10.1002/cpe.5402

WILEY

SPECIAL ISSUE PAPER

Machine learning algorithms to detect DDoS attacks in SDN

Reneilson Santos | **Danilo Souza** | **Walter Santo** | **Admilson Ribeiro** | **Edward Moreno**

PROCC, Federal University of Sergipe, São Cristóvão, Brazil

Correspondence
Reneilson Santos, PROCC, Federal University of Sergipe, Av. Marechal Rondon, s/n - Jardim Rosa Elze, São Cristóvão-SE 49100-000, Brazil.
Email: reneilson1@gmail.com

Summary

Summary Software-Defined Networking (SDN) is an emerging network paradigm that has gained significant traction from many researchers to address the requirement of current data centers. Although central control is the major advantage of SDN, it is also a single point of failure if it is made unreachable by a Distributed Denial of Service (DDoS) attack. Despite the large number of traditional detection solutions that exist currently, DDoS attacks continue to grow in frequency, volume, and severity. This paper brings an analysis of the problem and suggests the implementation of four machine learning algorithms (SVM, MLP, Decision Tree, and Random Forest) with the purpose of classifying DDoS attacks in an SDN simulated environment (Mininet 2.2.2). With this goal, the DDoS attacks were simulated using the Scapy tool with a list of valid IPs, acquiring, as a result, the best accuracy with the Random Forest algorithm and the best processing time with the Decision Tree algorithm. Moreover, it is shown the most important features to classify DDoS attacks and some drawbacks in the implementation of a classifier to detect the three kinds of DDoS attacks discussed in this paper (controller attack, flow-table attack, and bandwidth attack).

KEYWORDS

DDoS classification, decision tree, Mininet, MLP, random forest, SDN environment, SVM

1 | INTRODUCTION

Many difficulties that are found in current networks, such as vertical integration, coupling between the data and controller planes, difficulty to modify (or insert) the application into the network, and decentralization, helped in the emergence of a new network paradigm which would be able of eradicate these problems, the Software Defined Network (SDN).[1]

The great advantage of SDN is its capability to uncouple the data and controller planes, giving greater freedom to developers and facilitating the application maintenance in these environments.[2]

To uncouple the control plane and let it logically centralized, it was possible to abstract the network infrastructure from its applications. Thus, services implemented through hardware (eg, the IDS, Firewall, Routers) can be built using a software.[3] This centralization and uncoupling of SDN are allowed because the network controller and the switches have well-defined programmable interfaces[1] through APIs, which allow the communication between them. The OpenFlow is one of the most remarkable APIs for the SDN environment,[4] being one of the first APIs to be standardized.

The resources found in SDN allow the creation of components (applications) with low costs if compared with the creation of these in current networks (the costs in SDN are only related with the time to program). Therefore, SDN becomes an ideal environment to test new applications in a network.

Although the potential innovation is introduced in the use of SDN, it brings some new challenges on networks security.[5] These challenges can compromise the basic properties of a secure communication network such as confidentiality, integrity, availability of information, authentication, and nonrepudiation.[6]

According to Mousavi and St-Hilaire,[7] one of the most critical challenges in this type of environment is the Distributed Denial of Service (DDoS) attack. In this scenario, a large number of packets are sent to a host or group of hosts on a network. The collection of legitimate and counterfeit DDoS packages can bind the controller's capabilities to continuously process to the point where they are completely depleted. This will make the controller inaccessible to the legitimate newcomer packets and may bring down the controller, causing the loss of the SDN architecture.

On the other hand, detection of such DDoS attacks can be very difficult because of conventional detection methods since intruders can be distributed and located on different switches, and the detection process may not benefit from performing the detection process on the switches because the switches cannot detect attacks in a complete way.[8] In addition, DDoS attacks such as controller-switch communication flooding and switch flow-table flooding may affect considerably an SDN environment due to the centralized controller and flow-table limitation in the network device, which can make critical services unavailable.

In IoT environments, devices as controllers, sensors, and communication substrate, in critical infrastructures, can aggravate such problems.[9] Nonetheless, some solutions to SDN security challenges have already been proposed in the literature in recent years.[10-13] One technique that can be used to address these issues is to use machine learning algorithms.

The popularity of machine learning in recent years has seen the coupling of flow statistics with machine learning approaches to classify network traffic.[14] This approach is not new, despite its recent interest, a work published by Frank[15] already addressed the intrusion detection.

Machine learning has significant advantages over static and payload inspection-based techniques. The development of new approaches has been motivated by limitations in previous efforts. For instance, static classifiers are ineffective as they assume that applications use known port numbers that do not change.

In this context, this paper shows an IDS (intrusion detection system) technique to detect some DDoS attacks on an SDN environment with the implementation of four different machine learning algorithms (simplified here by ML-Algorithms). We have studied and analyzed (i) Multiple Layer Perceptron (MLP), (ii) Support Vector Machine (SVM), (iii) Decision Tree, and (iv) Random Forest algorithm, and we use the Scapy tool to generate the attacks and a Mininet with POX controller to simulate the SDN environment.

In order to analyze the potential of the ML-Algorithms' implementation on SDN (for the detection of DDoS attacks), this paper was divided as follows: in Section 2, the related works are presented, followed by Section 3 that presents briefly the theoretical foundation of DDoS attacks and detection techniques. After that, Sections 4 and 5 show, respectively, the definition and execution of the experimental planning. Results and conclusions are shown in the Sections 6 and 7, respectively.

2 | RELATED WORKS

In 2010, Braga et al[16] made a DDoS Flooding Attack Detector using SOM (Self Organizing Maps), a machine learning unsupervised algorithm. They work with a NOX controller, in which the switches of the network were monitored in predetermined periods. The detection works in three steps: the Flow Collector, the Feature Extractor, and the Classifier (SOM). In the experiment, seven different types of flooding attacks were made using the Stacheldraht tool to generate traffic. As a result, the SOM algorithm classifies the data in 271 ms with 4-tuple and 352 ms with 6-tuple (the tuple determines the number of features extracted), and in terms of accuracy, in all cases, the Detection Rate measurement was more than 98%.

In 2014, Kokila et al[13] proposed a protection scheme optimized for SDN against flood attacks based on the SVM classifier and a new algorithm called Idle-timeout Adjustment (IA). The authors analyzed the proposed scheme and evaluated the metrics in relation to the number of streams, the CPU usage of the SDN controller and of the OpenvSwitch.

Dao et al[17] created a method to track a type of DDoS attack that is worst in SDN than in normal networks (related to the time to process the first packet sent by a host). The method, therefore, is characterized by a table and a count variable to obtain the IPs that are from real users and block those that are malicious. The experiment is simulated using an OPNET modeler and the result is fewer entries on flow table, less bandwidth, and fewer packets on the controller. In addition, to solve this problem, Mousavi and St-Hilaire[7] created a method based on entropy, related to the destination IP address.

Tang et al[10] used MLP to detect intrusion in SDN, using six basic features easily obtained with OpenFlow functions. The MLP consists of three hidden layers, with the structure (12, 6, 3) and it is classified into two distinct classes, getting 75.75% of accuracy in the best case. The experiment was not performed in a real SDN environment, and despite the architecture being given, nothing besides that regarding the simulation environment was given.

Nanda et al[18] used machine learning algorithms, trained on historical network attack data, to identify the potential malicious connections and potential attack destinations. It was used four widely known machine learning algorithms, ie, C4.5 (Decision Tree), Bayesian Network (BN), Decision Table (DT), and Naive-Bayes (NB), to predict the host that will be attacked based on the historical data. Experimental results show that the average prediction accuracy of 91.68% is attained using Bayesian Networks. Still, in this paper, the authors used data from the LongTail project 19 to train the models. Leveraging the algorithm output, it is possible to define the security rules on the SDN controller to restrict the access of potential attackers by blocking the entire subnet.

Miao et al[19] proposed an analytical model based on the arrivals of the Poisson-Markov Process (MMPP), capturing real characteristics of the multimedia traffic patterns, in order to obtain a comprehensive and deep understanding of the performance behavior of the SDN network. The authors adopted a Priority-Queue (PQ) system to model the SDN data plane to capture the multiqueue nature of routing devices. In addition, Quality-of-Service performance metrics in terms of average latency and the average network throughput of SDN networks are derived based on the analytical model developed. The authors concluded that the validation results have revealed that the average latency and the average throughput predicted by the developed analytical model reasonably match those obtained from the simulation experiments.

Outside the SDN environment, many works can be found related to the use of classification algorithms to detect intrusion in networks. Dhanabal and Shantharajah,[11] in 2015, carried out a study about some techniques and compared them, using the NSL-KDD as the dataset, based on the KDD99 dataset, but with some improvements. The techniques used are J48 (a Decision Tree algorithm), SVM, and Naive Bayes; and for the DoS attack, the J48 was better than the other ones. Still regarding this matter, in 2015, Pervez and Farid[12] used the SVM with a different number of features and had good results (ie, 99% of accuracy with all features of NSL-KDD).

Finally, Wang et al[20] introduced the concept of a network partition and proposed a cluster-based partitioning algorithm called Clustering-based Network Partition Algorithm (CNPA). Network partitioning is intended to select appropriate controller locations to shorten the latency between controllers and switches in long-distance networks. CNPA can ensure that each partition is able to shorten the maximum latency between controllers and switches.

3 | DDOS ATTACKS IN SDN

The purpose of this section is to enlighten background information about the DDoS attacks in the SDN environment. First, we briefly describe the DDoS attacks. This is followed by a description of the DDoS attack defensive mechanisms.

3.1 | DDoS attacks

Distributed Denial of Service (DDoS) attacks have been a real threat in many aspects of computer networks and distributed applications. The main objective of a DDoS attack is to bring down the services of a target using multiple sources that are distributed. For example, attackers can transfer thousands of packets to a victim to overwhelm its access bandwidth with illegitimate traffic, making online services unavailable. There are numerous denials of service (DoS) attack methods being used to degrade the performance or availability of targeted services on the Internet.[5] Usually, these methods can be classified as challenges associated with the SDN at each layer of the framework, application, control, and infrastructure.

3.1.1 | Application layer DDoS attacks

Application layer attacks use the software in a malicious way, aiming to exhaust resources to process any further requests. These attacks are generally harder to detect on the network level as they show no clear deviation from legitimate traffic.[5]

Since the isolation of applications or resources in SDN is not well solved, DDoS attacks on one application can also affect other applications. A common example is the HTTP flooding attacks.[21]

3.1.2 | Control layer DDoS attacks

Controllers of SDN and their communications can be subjected to different types of attacks.[2] Among the threats that can cause significant damages are the following: attacks on the control plane and communication between the controller and other networks components, eg, northbound API, southbound API, westbound API, or eastbound API. In addition, the controller can be considered a single point of failure and scalability that raises potential performance problems and unavailability of the control plane.

3.1.3 | Infrastructure layer DDoS attacks

Infrastructure layer DDoS attacks could potentially overload through two points: switches or by attacking the southbound API.[22] For example, huge traffic may be sent by an attacker to execute a DoS attack on the node by setting up a number of new and unknown flows infrastructure layer.

3.2 | DDoS attack detection techniques

DDoS attacks have been studied for a long time and the types of threats to them are mostly known.[5] However, SDN is a new architecture and the studies are at an early stage. In addition, SDN networks have distinct detection methods for different types of DDoS attacks.[23] These methods include entropy-based,[7,24-26] machine learning–based,[10-13] traffic pattern analysis,[27] connection rate,[27,28] and techniques that combine the use of the IDS and OpenFlow.[29-31]

3.2.1 | Entropy

The ability to measure randomness in packets arriving on a network makes entropy-based methods good candidates for the DDoS detection. The greater the randomness, the greater the entropy, and vice versa. Entropy-based methods depend on network feature distributions to detect

anomalous network activities.[24] The presence of anomalies in an SDN network can be identified by adopting the use of predefined thresholds. In addition, probability distributions of various network features such as source IP address, destination IP address, and port numbers are used to calculate the entropy.[32]

3.2.2 | Machine learning

Machine learning–based methods employ techniques to detect anomalies in a network environment, these can be based on models, based on statistic and math, based on unsupervised machine learning algorithms, and based on supervised machine learning algorithms.[33] These algorithms take into account various network features and traffic characteristics to detect the presence of anomalies.

In fact, any system that is built to detect any anomalies in the network catch the traffic on it and extract some kind of information from them, and the approach that uses machine learning is trying to catch the pattern of normal and abnormal traffic on the network, without the need to know the pattern itself.

3.2.3 | Traffic pattern analysis

These techniques work on the assumptions that the infected hosts exhibit similar behavioral patterns that are different from benign hosts.[34] Therefore, it analyzes the traffic relating to the metrics of the attack pattern networks in order to identify the attacker or the target under attack. Patterns are observed as a result of a command that is sent to many members of the same botnet causing a similar behavior (eg, sending illegitimate packets, starting to scan).

3.2.4 | Connection rate

Bawany et al[32] define connection rate techniques as ''the probability of a connection attempt being successful should be much higher for a benign host than a malicious host.'' Whenever the likelihood ratio for a host to exceed a certain threshold, it is declared as infected. These techniques are classified into two types: connection success ratio and connection rate, which refers to the number of connections instantiated within a certain window of time.

3.2.5 | SNORT and OpenFlow integrated

Recent researches have combined the use of an intrusion detection system and OpenFlow to detect attacks and reconfigure the network dynamically.[30,32] An intrusion detection system (IDS) monitors the traffic to identify malicious activities. OpenFlow switches are then dynamically reconfigured based on the detected attacks in real time.

The information presented in the current section gives an overview of the DDoS attacks and DDoS detection techniques in an SDN environment. In the next section, we present the definition and execution of the experiment planning.

4 | DEFINITION AND EXPERIMENT PLANNING

In this section, following the methodology described in the work of Wohlin et al,[35] we present the experimental planning related to the DDoS attack detection in the SDN environment. The next subsections explain in details the definition of the experiment and its particularities, such as the goals and the planning of the experiment (so, we present the context, show the hypothesis, the project, and what is used to achieve the goals and complete the experiment).

4.1 | Goal definition

The main goal of the experiment is to analyze ways to solve the problem of the DDoS attacks in the SDN environment. Moreover, a secondary goal is to compare different ML-Algorithms as a solution to the problem, analyzing metrics as accuracy and time to process in the SDN environment.

Following the formalization of the GQM model, proposed in the work of Basili et al,[36] both objectives can be unified as follows: **To analyze** security methods to identify DDoS attacks in SDN, **with the purpose of** comparing them **with respect to** efficacy (accuracy) and efficiency (time to process) **from the point of view of** researchers and developers of SDN and Information Security **in the context of** a simulated DDoS attack on a Mininet Virtual Network with a POX controller (Mininet VN).

4.2 | Planning

Context Selection: the experiment was *in vitro*, in which the DDoS attack and normal traffic were simulated in the Mininet VN. In addition, to detect the anomalies and the normal traffic, four different ML-Algorithms were used (MLP, SVM, Decision Tree, and Random Forest).

TABLE 1 Hyperparameters and ML-Algorithms

Models	Hyperparameters
MLP	Activation function
	Weight update function
	Hidden layers
	L2 regularization (α)
SVM	Kernel
	Regularization factor (C)
	Classification approach
Decision Tree (CART)	Selection criteria
	Presort
	Maximum number of features
	Splitter
Random Forest	Number of trees
	Selection criteria
	Maximum number of features

TABLE 2 Description of studied features

Number	Feature	Description
0	*byte_count*	Number of bytes in flow
1	*cookie*	Opaque controller-issued identifier
2	*dl_dst*	Ethernet destination port
3	*dl_src*	Ethernet source address
4	*dl_type*	Ethernet frame type
5	*dl_vlan*	Ethernet VLAN ID
6	*dl_vlan_pcp*	Ethernet VLAN priority
7	*duration_nsec*	Time flow has been alive in nanoseconds
8	*duration_sec*	Time flow has been alive in seconds
9	*hard_timeout*	Max time before discarding (seconds)
10	*idle_timeout*	Idle time before discarding (seconds)
11	*in_port*	Port ID
12	*max_len*	Max length to send to controller
13	*nw_dst*	IP destination address
14	*nw_proto*	IP protocol
15	*nw_src*	IP source port
16	*nw_tos*	Type of service
17	*packet_count*	Number of packets in flows
18	*priority*	Priority level of flow entry
19	*port*	Output port
20	*table_id*	ID of the table to put the flow in
21	*tp_dst*	TCP Destination Port
22	*tp_src*	TCP Source Port
23	*type*	Type of action

Dependent Variables: accuracy and the processing time of each ML-Algorithm, besides their set of hyperparameters that can also be considered as dependent variables because each hyperparameter depends on the data used to train. Table 1 shows each hyperparameter used in this experiment associated with the respective machine learning model.

Independent Variables: DDoS attacks and their parameters (presented in Table 2), the POX Component (implemented by the authors), and the ML-algorithms.

Hypothesis Formulation: the research questions to this experiment are the following: could the problem with DDoS attacks be identified or mitigated using a component in the SDN environment? Could some of the ML-Algorithms (MLP, SVM, Decision Tree or Random Forest) be considered better than all the others in terms of efficacy and efficiency?

According to these questions, the hypothesis that could be verified are presented as follows (to know, DT = Decision Tree, and RF = Random Forest):

Hypothesis 1.

H0: *It is not possible to implement a component in SDN that can identify normal traffic and anomalies (specifically DDoS attacks) in a feasible way (time superior to 0.5 seconds and the accuracy is less than 50%).*

H1: *There is a way to identify and classify the DDoS attacks and normal traffic using ML-Algorithms implemented as a component on the controller in the SDN environment (time inferior to 0.5 seconds and accuracy superior to 50%).*

Hypothesis 2.

H0: *There is no difference among the metrics of the algorithms* $\mu_{SVM} = \mu_{MLP} = \mu_{DT} = \mu_{RF}$.

H1: *Some algorithms have different metrics related to the others.* $(\exists \mu_{alg_x} \neq \mu_{alg_y})$.

As mentioned before, the metrics used in this paper were the time to process (time to classify between anomaly or normal traffic) and accuracy.

Objects Selection: the experiment used DDoS attacks which were simulated varying the IP source (list with more than 20 000 IPs), port destination (the choice depends on the application protocol or, when just a TCP packet or a UDP packet is sent, the port was chosen randomly), IP destination (six hosts connected to the SDN environment), Transport Protocol (TCP and UDP), and Application Protocol (ICMP, HTTP, SMTP).

All the features are captured on the flow table of the OpenFlow switch simulated on the Mininet VN. Initially, some effort was made to choose the best set of features to classify the anomalies; however, the results showed that using all the features, without a long preprocessing step, the classification was expedited and did not significantly influence the results, therefore it was chosen to maintain all of them together.

Experiment Project: every model used the same data (stored in a file) to evaluate the classification in terms of efficacy. In addition, all the DDoS attacks were generated randomly varying the independent variables, using the Scapy tool (http://www.secdev.org/projects/scapy/). Table 3 shows each traffic generated by each simulator.

In this way, we covered three different categories of vulnerabilities in the SDN environment. In Figures 1, 2, and 3, we can visualize the traffic behavior on each of these three cases.

- Controller attack (Figure 1)
- Flow-table attack (Figure 2)
- Bandwidth between switch and controller attack (Figure 3)

In Figure 4, we can observe the normal traffic on the SDN environment. The controller attack was made using the TCP SYN Flood, in which the Switch received a packet with a new rule, sent to the controller to create the rule for this packet. With a lot of new rules arriving at the controller, it is overloaded and crashes. In this case, it was used 20 000 different IPs saved in a list and chosen randomly to avoid bias on the experiment.

The flow-table attack was made using HTTP flooding, ICMP flooding, and UDP flooding, also using the same list of IPs used in the controller attack. In this case, each "client" was making, at most, 10 requests to the servers (the hosts on the Mininet VN), then the table on the switch will increase until there is no more space to new rules, and as a consequence, the time to respond to each new requisition increased.

Finally, the bandwidth attack was made using only the HTTP Flooding, in this case, the size of the IP packet was increased and sent to the hosts; the same list of IPs was used, but in this case, the size of the packets was more important than the number of clients.

TABLE 3 Traffic simulation using scapy tool

Simulator	Attacks used
Scapy	Normal Traffic
	ICMP Flooding
	HTTP Flooding
	TCP SYN Flood
	UDP Flood

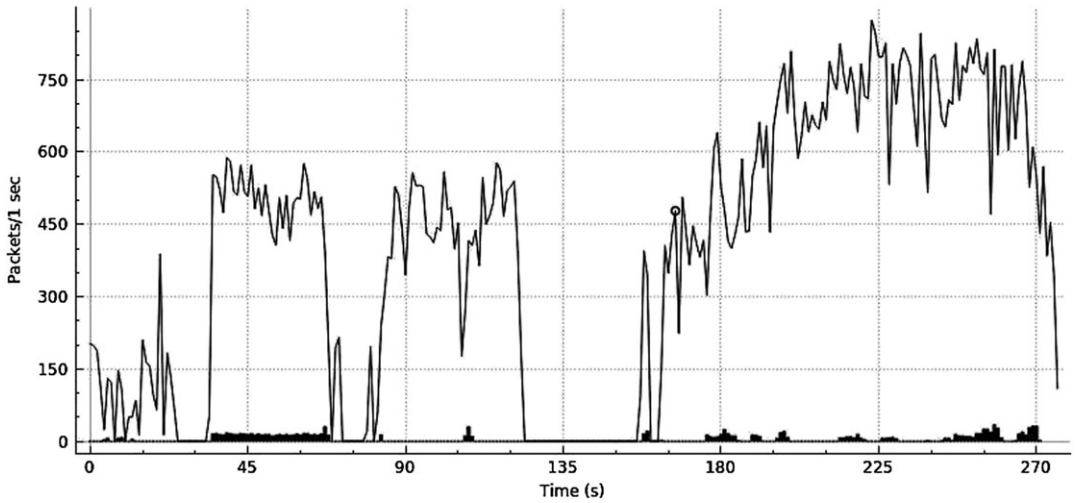

FIGURE 1 Controller attack - network traffic

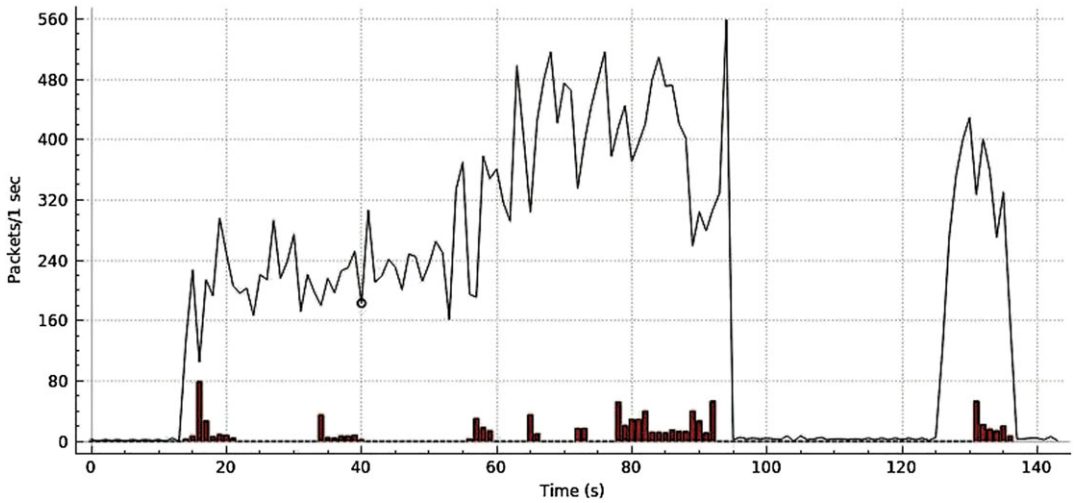

FIGURE 2 Flow-table attack - network traffic

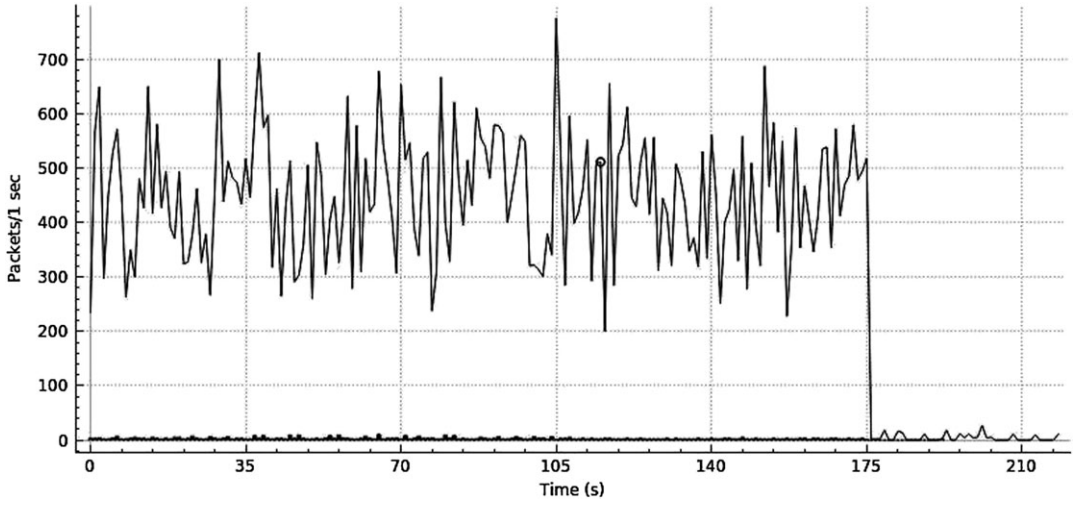

FIGURE 3 Bandwidth attack - network traffic

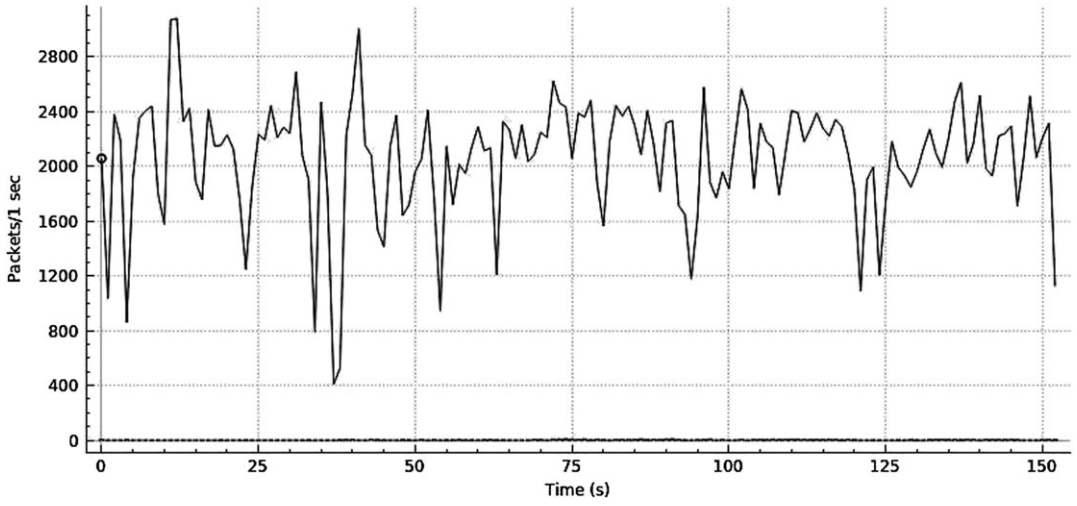

FIGURE 4 Normal network traffic

In all cases, six different clients were used to send data simultaneously for the target hosts created in the Mininet environment.
In the next section, an image of the SDN architecture for this experiment is shown in Figure 5.
Besides that, the models are implemented directly on the Mininet VN to get the efficiency metric (processing time) of each model.

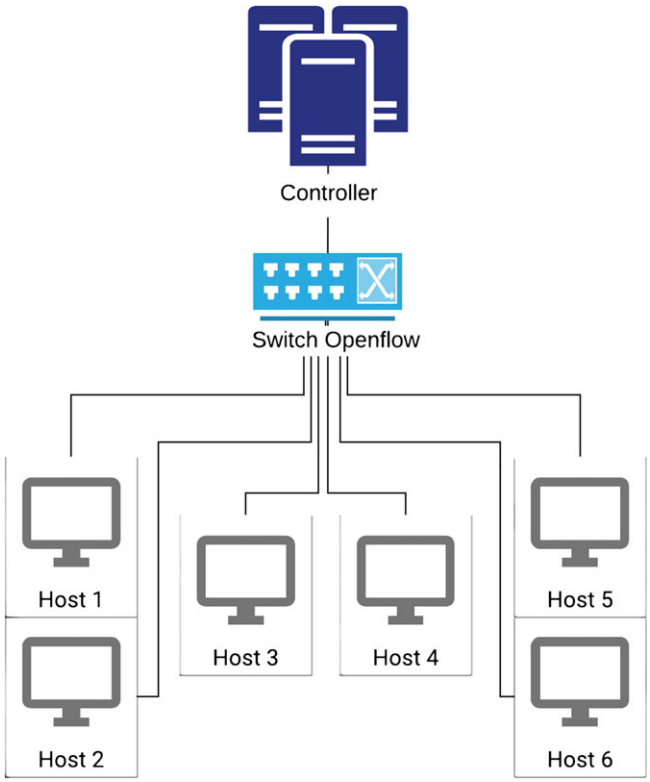

FIGURE 5 Mininet architecture illustration

Instrumentation: the algorithms were obtained through Python libraries at the 3.5.1 version. To get the best hyperparameters, the algorithms were executed in an HP desktop, with 4 Gb of RAM, Intel i5-3470S (2.9 GHz) executing in an Ubuntu 12.04.5 LTS operational system. The acquisition of the metrics to verify the hypothesis was made in a Virtual Machine, with 1 Gb of RAM, Intel i5-3470S (2.9 GHz) executing in a Mininet 2.2.2 on Ubuntu 14.04 LTS - 32 bit, and the controller is the POX.

5 | EXPERIMENT EXECUTION

5.1 | Preparation

According to Giotis et al,[23] DDoS attacks on the SDN environment have new challenges to deal with (attack on the controller, attack on the dataflow table, and attack on the bandwidth between the switch and the controller are some examples), in addition to the problems already known about this type of attack on common network environments. Despite these new challenges in SDN, this new technology also allows new solutions, using the dataflow captured from the switch and deal with it in the controller (implementing a new solution in software).

Therefore, in this paper, the Scapy tool is used to generate the traffic flow (normal and malicious), using different protocols, different IP sources, different port numbers, and some others variables which were randomly chosen. We generated the three examples of new challenges related to DDoS attacks on the SDN environment (attack on the controller, attack on the dataflow table, and attack on the bandwidth), already detailed on the last section.

The experiment is performed using Mininet VN by contextualizing a Smart Home IoT network. This scenario is shown in Figure 5, with six hosts (simulating the IoT components connected to the SDN), one switch, and one controller.

After the construction of the environment in the Mininet VN, the next step is to implement a POX component to capture the dataflow on the switch OpenFlow and store the captured data in a file. This file is used as input by the algorithms in both the training phase and test data. Finally, the last step was to implement the component to detect the DDoS attack with the ML-Algorithms already trained on the previous step and observe its behavior on the simulated SDN environment.

5.2 | Data collection

To collect the data, a specific POX component was codified in the SDN environment to store in a file of the dataflow. Firstly, in six clients, the Scapy tool was used to generate malicious data traffic. Secondly, to obtain traffic in the network, using six clients again, the "ping" command and the Scapy tool were used, generating HTTP and ICMP normal traffic. In both cases (malicious and normal traffic), the six hosts of the simulated environment were the target.

In the end, 70% of the dataset (containing 20 000 data of the dataflow table, being 10 000 for each type of traffic) was used to evaluate the hyperparameters of the models. The *GridSearch* technique, implemented by the *Scikit-learn*,[37] a Python machine learning library, allowed the local search of the best hyperparameters set. In order to avoid model overfitting (especially for SVM and MLP models), standardization of the features values was applied to the data.

It is important to remember that the 10 000 data used as malicious were captured from three different kinds of attacks (3333 as attack on controller, 3333 as attack on dataflow table, and 3334 as attack on bandwidth).

Once the best hyperparameters set for each model was achieved, the accuracy was obtained using the test dataset (30% of the dataset randomly chosen).

After that, to get the efficiency metric, the models were implemented in the Mininet VN and the data were obtained by the dataflow table in each 2 seconds and processed by the model, calculating the differential time between the start and the end of the execution phase on classification model. In this part of the experiment, the data (processing time) were saved in a file (a total of 50 different acquisitions) to be analyzed *a posteriori* using statistical analysis.

Moreover, to test the accuracy for each kind of attack, three other data sets were created to achieve this efficacy metric, each one with 10 000 of normal attack and 10 000 of some other attack (controller, bandwidth, and flow table), in this way, it is possible to analyze the efficacy of the ML-Algorithms for each kind of DDoS attack. Finally, all the data acquired were validated as shown in the following subsection.

5.3 | Data validation

In the training step, the algorithm used the K-fold technique[38] avoids algorithms' overfitting. Related to the efficiency metric, the Kolmogorov-Smirnov test (KS-Test) was used to analyze if the data follows a normal distribution, what is refuted by a low p-value (close to zero), indicating that the processing time does not follow a normal distribution for all models (Random Forest, Decision Tree, SVM, and MLP).

After that, these data were analyzed using the Friedman test to verify the hypothesis raised and the result was that there are statistical differences in the mean of the processing time of these algorithms, just the MLP and the Decision Tree have no statistical difference according to the result of the test. In the next section, the results and a discussion about them are presented in more details.

6 | RESULTS

Table 4 shows the best hyperparameters set for each ML-Algorithm.

Before starting to analyze the accuracies graphs, it is important that the hyperparameters were selected and understand the importance of each feature shown in Table 2 for the classification process. Figure 6 shows the importance of each feature, the numbers shown on the figure are the same as presented in Table 2.

Continuing with the accuracies, Figure 7 shows the total accuracy (given by Equation (1)) of the machine learning algorithms for each attack simulated.

$$Accuracy = \frac{Number\ of\ Correct\ Classifications}{Total\ of\ Samples} \tag{1}$$

The accuracy is a statistical value that determines how close our ML-Algorithm is to the ideal. If this value is 1, it means that the algorithm has no error and classifies the data perfectly. This is a good metric when the data are equally distributed (the same quantity of data for each class), as is the case of our experiment.[38]

TABLE 4 The best hyperparameters set for each ML-Algorithm

Models	Hyperparameters
MLP	Activation function: Relu Weight update function: Quasi-Newton method (*lbfgs*) Layers: 1 hidden (15 neurons) L2 regularization (α): 1e-3
SVM	Kernel: RBF Regularization Factor (C): 1e+5 Classification approach: One against rest (*ovr*)
Decision Tree (CART)	Selection criteria: Gini Presort: True Maximum number of features: $\sqrt{Features}$ Splitter: best
Random Forest	Number of trees: 25 Selection criteria: Entropy Maximum number of features: $\sqrt{Features}$

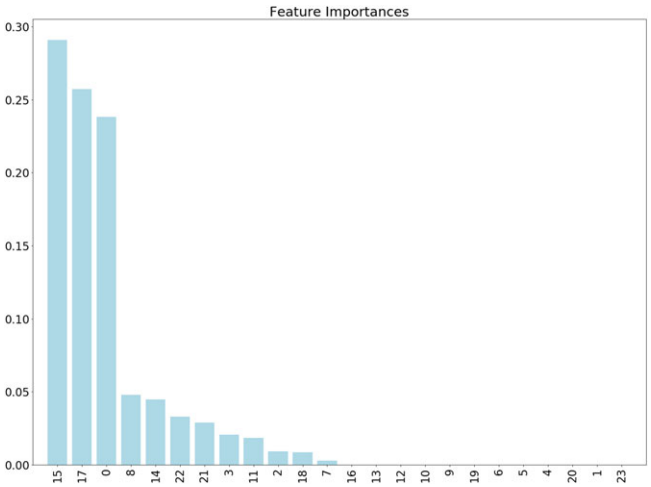

FIGURE 6 Importance of each feature

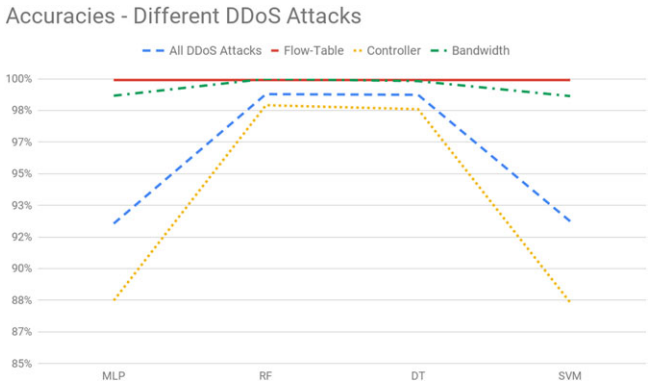

FIGURE 7 Accuracies of ML-Algorithms for each DDoS attack

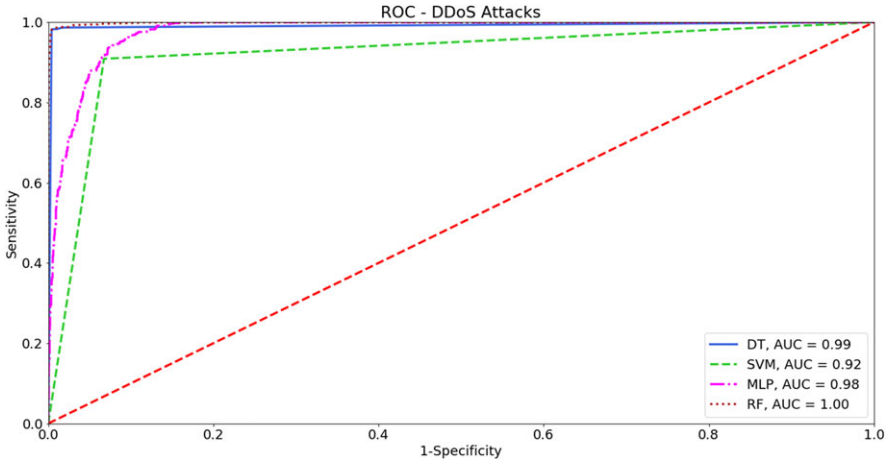

FIGURE 8 ROC curve - all DDoS attacks

Figure 8 shows the ROC curve for the classification of all the three kinds of attack simulated.

In addition, Figure 9, Figure 10, and Figure 11 show the ROC curve for each kind of DDoS attack simulated in this experiment (attack on controller, attack on the flow table, and attack on the bandwidth between the controller and switch overflow, respectively).

The ROC curve represents a relation between sensitivity (in our experiment represented by the percentage of data classified as malicious that is really malicious) and 1 – *specificity* (in our experiment, the specificity is represented by the percentage of data classified as malicious that is really malicious, then 1 minus this value is the opposite, ie, the percentage of data classified as normal but that are malicious).[38]

This curve is very used in the machine learning to choose a good point for the classifiers, given by the point above the central curve in which the distance between them is maximum.

In the graphs, we also show the AUC metric, that is, the area under the curve. When this metric is higher, then the classification is better. Therefore, that is a good metric to evaluate any ML-Algorithm that classifies a binary system (as is the case of our experiment, malicious vs normal traffic).

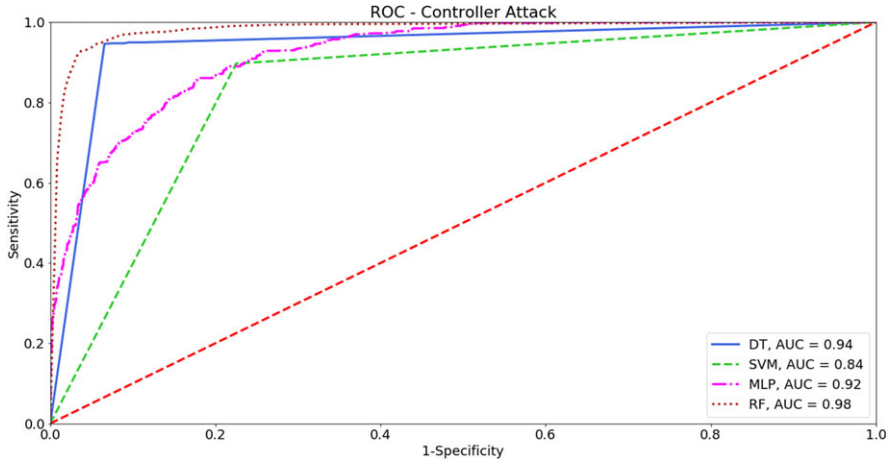

FIGURE 9 ROC curve - controller attack

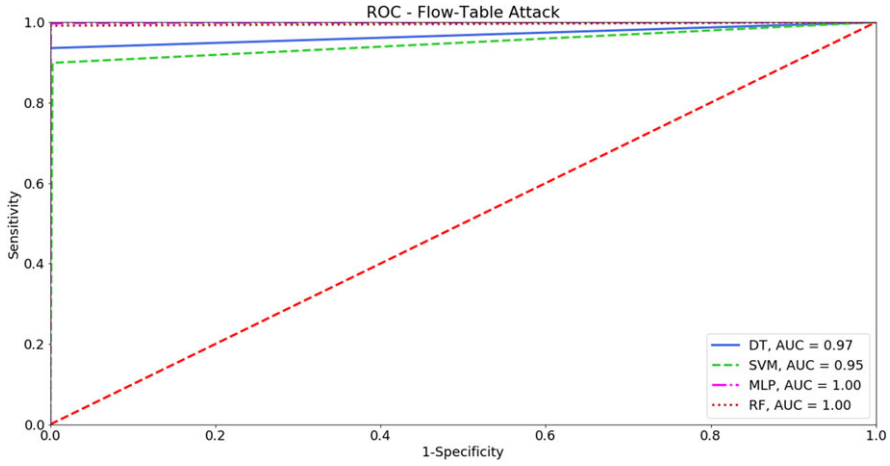

FIGURE 10 ROC curve - flow-table attack

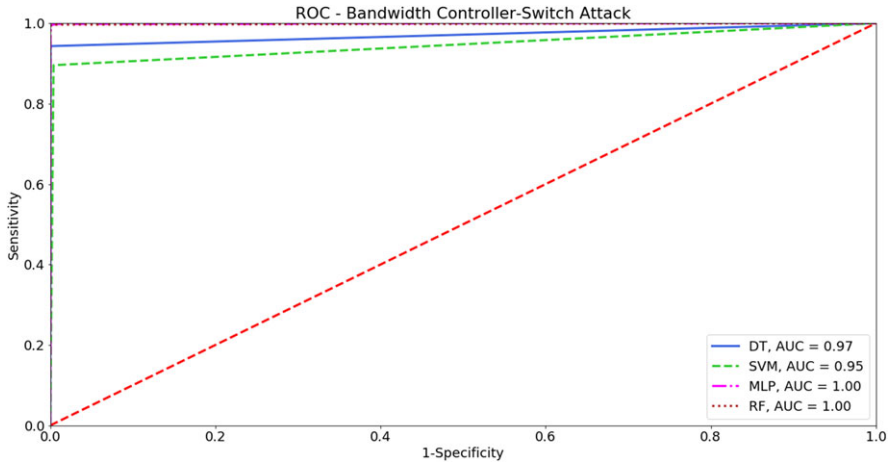

FIGURE 11 Bandwidth controller-switch attack

Finally, as the result of the time to process each algorithm, Figure 12 shows the algorithm behavior in the classification of DDoS simulated attacks on the SDN.

6.1 | Analysis and interpretation

As explained in the previous section, the algorithms were trained using the K-fold technique[38]; in this way, it avoided the overfitting of the classifiers. Regarding that, the results shown in Figure 7 are related to the 30% of the data separated for test, in other words, 70% of the data were used as training data, in which the K-fold technique was applied with K=10, and the others 30% of the data were used to test the algorithms, this process is needed to confirm if the algorithms were not overfitted.

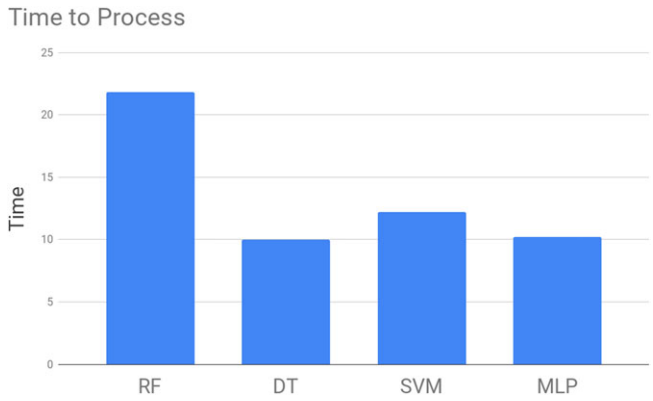

FIGURE 12 Time to process

TABLE 5 Important features for classification

Position	Feature	Description
1st	*nw_src*	IP source port
2nd	*duration_nsec*	Time flow has been alive in nanoseconds
3rd	*in_port*	Port ID
4th	*dl_src*	Ethernet source address
5th	*duration_sec*	Time flow has been alive in seconds
6th	*byte_count*	Number of bytes in flow
7th	*packet_count*	Number of packets in flows
8th	*tp_dst*	TCP destination port
9th	*tp_src*	TCP source port
10th	*nw_proto*	IP protocol
11st	*dl_dst*	Ethernet destination port

As seen in Figure 6, there are some features that have no importance on the classification process. Table 5 shows the features according to its importance for the detection of the anomalies.

All the features with less importance (sometimes not used for the algorithms) are, in general, with a few variations on their values. By contrast, the IP Source Port (nw_src) is the most important feature in the classification, followed by the duration time (in nanoseconds) in which the connection is alive. This result is important because it could help to build a DDoS mitigation system.

As a result of all kinds of attacks, both the Random Forest algorithm and the Decision Tree algorithm have good results in terms of efficacy, with almost 100% of accuracy. However, as we can see in Figure 12, the Decision Tree algorithm had the lowest processing time, whereas the Random Forest algorithm had the largest one, knowing that, in this context of a DDoS attacks, it is more important take in consideration the time to process the algorithm and, as a consequence, it is possible to say that the more suitable algorithm for this problem is the Decision Tree.

These algorithms are better for this problem because they have intrinsically a process to choose the best parameters that should be used on the classification process, using (in our case) the entropy function with this purpose. The MLP and the SVM are also good algorithms; however, as the result of these algorithms are more dependently of mathematical operations (according to multiplications and sums of the feature vectors and some specific parameters of each algorithm), they are not the best classifiers for this problem, even with the normalization of the features.

Moreover, related to the attacks analyzed separately, it is possible to notice that the flow-table attack is the one with the best metrics of efficacy in the classification process, whereas the controller attack has the worst. In addition, it is possible to observe that the Random Forest algorithm has always the best result in all three cases.

This result (the controller attack with the worst environment to be classified by all the algorithms, with values less than 90% of accuracy for MLP and SVM) happens because the TCP SYN Flood was used as the attack in the controller, and in this case, the time flow, in nanoseconds, in which the connection is alive, is quite equal to the normal traffic. Therefore, if this feature is important in the detection of the other types of attacks, in the case of the controller attack, this makes the classification worst, and, as the classifiers are trained with the database equally distributed, the flow-table attack and the bandwidth attack have better results.

Additionally, the problem with these algorithms is, when a new type of attack happens to the network, they will not capture them. In other words, the implemented algorithms were not online, in which the new information obtained and identified as a DDoS attack by a specialist improves the model. This type of classifier is intended to be implemented in a future work, even because as explained in Section 4, the main goal of this paper was to prove that it is possible to identify the new kinds of DDoS attack in the SDN environment using algorithms already known in the literature, which is confirmed by the good results achieved.

As a conclusion, we can say that using a Decision Tree algorithm in an SDN environment improves the security without big losses on the traffic of the network because this is a ‘‘light’’ algorithm, with no high processing power or memory consumption ideal for the network environment.

6.2 | Threats to validity

A K-fold algorithm was used as a way to mitigate biases related to concluding remarks in regards to the established hypothesis (**conclusion validity**). Besides that, still related to **conclusion validity** and **external validity**, a database with attacks randomly generated and with a big list of IP sources was used.

A particular threat to validity is the fact that all traffic has been generated artificially (**conclusion validity**); however, as a way to be near to the real-world network, some variables have been randomly generated (or chosen), like IP, protocols, size of packets, and quantities of requisitions by only one "client."

7 | CONCLUSION

The emergence of SDN environments, ie, the possibility to make our own applications to mitigate security issues or to propose new software solutions on the network environment, has made many researchers and industries look favorably on this new concept.

However, together with the new possibilities to build solutions, SDN also brings some new problems to be treated. One of these problems is new kinds of DDoS attacks. For example, the DDoS attack in the SDN environment could attack the centralized controller and put down all the network if the controller crashes.

In this work, the ML-Algorithms (MLP, SVM, Decision Tree, and Random Forest) were proposed to detect DDoS attacks in three different categories (flow-table attack, bandwidth attack, and controller attack). Every attack was made using the Scapy tool, using a list with more than 20 000 IPs used as attackers, and the hosts connected to the SDN network as targets, simulated using the Mininet VN.

As a result of the experiment, the Decision Tree was found to be the best in general, mainly because it has the lowest time to process (although the Random Forest has had the best accuracy in absolute value). Moreover, it is possible to see that the Port Source is the most important feature in the classification of the anomalies, followed by the duration time (in nanoseconds) in which the connection is alive, what can help to build the mitigation DDoS system.

In addition, another result shows that, because the equal distribution of the database is used to train the classifiers, the controller attack has the worst classification accuracy. Therefore, if the developer is more concerned with this kind of attack, a specific classifier could be built to classify it, from which other features can be chosen to better classify this kind of DDoS Attack.

As a conclusion, in future works, we intend to use real traffic and implement a technique to mitigate the DDoS attacks or avoid them as soon as possible. In addition, a new experiment in a prototyped SDN environment using the Raspberry Pi platform is also an intended future work, which will provide more support to the proposed solution.

ACKNOWLEDGMENT

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, CNPq, and FAPITEC.

ORCID

Reneilson Santos https://orcid.org/0000-0001-9328-9744

REFERENCES

1. Kreutz D, Ramos FMV, Verissimo PE, Rothenberg CE, Azodolmolky S, Uhlig S. Software-defined networking: a comprehensive survey. *Proc IEEE*. 2015;103(1):14-76.
2. Wang B, Zheng Y, Lou W, Hou YT. DDoS attack protection in the era of cloud computing and software-defined networking. *Computer Networks*. 2015;81:308-319.
3. Sezer S, Scott-Hayward S, Chouhan PK, et al. Are we ready for SDN? Implementation challenges for software-defined networks. *IEEE Commun Mag*. 2013;51(7):36-43.
4. McKeown N, Anderson T, Balakrishnan H, et al. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Comput Commun Rev*. 2008;38(2):69-74.
5. Scott-Hayward S, O'Callaghan G, Sezer S. SDN security: a survey. Paper presented at: 2013 IEEE SDN for Future Networks and Services (SDN4FNS); 2013; Trento, Italy.
6. Douligeris C, Serpanos DN. *Network Security: Current Status and Future Directions*. Hoboken, NJ: John Wiley and Sons; 2007.
7. Mousavi SM, St-Hilaire M. Early detection of DDoS attacks against SDN controllers. Paper presented at: 2015 International Conference on Computing, Networking and Communications (ICNC); 2015; Garden Grove, CA.
8. Oppenheim AV. *Sinais e Sistemas*. São Paulo, Brazil: Prentice-Hall; 2010.
9. Özçelik M, Chalabianloo N, Gür G. Software-defined edge defense against IoT-based DDoS. Paper presented at: 2017 IEEE International Conference on Computer and Information Technology (CIT); 2017; Helsinki, Finland.

10. Tang TA, Mhamdi L, McLernon D, Zaidi SAR, Ghogho M. Deep learning approach for network intrusion detection in software defined networking. Paper presented at: 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM); 2016; Fez, Morocco.
11. Dhanabal L, Shantharajah S. A study on NSL-KDD dataset for intrusion detection system based on classification algorithms. *Int J Adv Res Comput Commun Eng*. 2015;4(6):446-452.
12. Pervez MS, Farid DM. Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs. Paper presented at: The 8th International Conference on Software, Knowledge, Information Management and Applications (SKIMA); 2014; Dhaka, Bangladesh.
13. Kokila RT, Selvi ST, Govindarajan K. DDoS detection and analysis in SDN-based environment using support vector machine classifier. Paper presented at: 2014 Sixth International Conference on Advanced Computing (ICoAC); 2014; Chennai, India.
14. Zhang J, Chen X, Xiang Y, Zhou W, Wu J. Robust network traffic classification. *IEEE/ACM Trans Networking*. 2015;23(4):1257-1270.
15. Frank J. Artificial intelligence and intrusion detection: current and future directions. In: Proceedings of the 17th National Computer Security Conference; 1994; Baltimore, MD.
16. Braga R, Mota E, Passito A. Lightweight DDoS flooding attack detection using NOX/OpenFlow. Paper presented at: IEEE Local Computer Network Conference; 2010; Denver, CO.
17. Dao NN, Park J, Park M, Cho S. A feasible method to combat against DDoS attack in SDN network. Paper presented at: 2015 International Conference on Information Networking (ICOIN); 2015; Cambodia.
18. Nanda S, Zafari F, DeCusatis C, Wedaa E, Yang B. Predicting network attack patterns in SDN using machine learning approach. Paper presented at: 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN); 2016; Palo Alto, CA.
19. Miao W, Min G, Wu Y, Wang H, Hu J. Performance modelling and analysis of software-defined networking under bursty multimedia traffic. *ACM Trans Multimed Comput Commun Appl*. 2016;12(5s). Article ID 77.
20. Wang G, Zhao Y, Huang J, Wu Y. An effective approach to controller placement in software defined wide area networks. *IEEE Trans Netw Serv Manag*. 2018;15(1):344-355.
21. Wang C, Miu TT, Luo X, Wang J. SkyShield: a sketch-based defense system against application layer DDoS attacks. *IEEE Trans Inf Forensics Secur*. 2018;13(3):559-573.
22. Yan Q, Yu FR. Distributed denial of service attacks in software-defined networking with cloud computing. *IEEE Commun Mag*. 2015;53(4):52-59.
23. Yan Q, Yu FR, Gong Q, Li J. Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: a survey, some research issues, and challenges. *IEEE Commun Surv Tutor*. 2016;18(1):602-622.
24. Wang R, Jia Z, Ju L. An entropy-based distributed DDoS detection mechanism in software-defined networking. Paper presented at: 2015 IEEE Trustcom/BigDataSE/ISPA; 2015; Helsinki, Finland.
25. Giotis K, Argyropoulos C, Androulidakis G, Kalogeras D, Maglaris V. Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments. *Computer Networks*. 2014;62:122-136.
26. da Silva AS, Wickboldt JA, Granville LZ, Schaeffer-Filho A. ATLANTIC: a framework for anomaly traffic detection, classification, and mitigation in SDN. Paper presented at: 2016 IEEE/IFIP Network Operations and Management Symposium (NOMS); 2016; Istanbul, Turkey.
27. Shin SW, Porras P, Yegneswara V, Fong M, Gu G, Tyson M. Fresco: modular composable security services for software-defined networks. Paper presented at: 20th Annual Network and Distributed System Security Symposium; 2013; San Diego, CA.
28. Dotcenko S, Vladyko A, Letenko I. A fuzzy logic-based information security management for software-defined networks. Paper presented at: 16th International Conference on Advanced Communication Technology; 2014; Pyeongchang, South Korea.
29. Li HC, Wa P. Implementation of an SDN-based security defense mechanism against DDoS attacks. Paper presented at: 2016 International Conference on Economics and Management Engineering (ICEME) and International Conference on Economics and Business Management (EBM); 2016; Wuhan, China.
30. Martins JS, Campos MB. A security architecture proposal for detection and response to threats in SDN networks. Paper presented at: 2016 IEEE ANDESCON; 2016; Arequipa, Peru.
31. Ha T, Kim S, An N, et al. Suspicious traffic sampling for intrusion detection in software-defined networks. *Computer Networks*. 2016;109:172-182.
32. Bawany NZ, Shamsi JA, Salah K. DDoS attack detection and mitigation using SDN: methods, practices, and solutions. *Arab J Sci Eng*. 2017;42(2):425-441.
33. Lazarevic A, Ertoz L, Kumar V, Ozgur A, Srivastava J. A comparative study of anomaly detection schemes in network intrusion detection. In: Proceedings of the 2003 SIAM International Conference on Data Mining; 2003; San Francisco, CA.
34. Harang R. Bridging the semantic gap: human factors in anomaly-based intrusion detection systems. In: *Network Science and Cybersecurity*. New York, NY: Springer Science+Business Media; 2014:15-37.
35. Wohlin C, Runeson P, Höst M, Ohlsson MC, Regnell B, Wesslén A. *Experimentation in Software Engineering*. Berlin, Germany: Springer Science+Business Media; 2012.
36. Basili VR, Caldiera G, Rombach HD. The goal question metric approach. In: *Encyclopedia of Software Engineering, Volume 2*. 1994:528-532.
37. Pedregosa F, Varoquaux G, Gramfort A, et al. Scikit-learn: machine learning in python. *J Mach Learn Res*. 2011;12:2825-2830.
38. Michalski RS, Carbonell JG, Mitchell TM. *Machine Learning: An Artificial Intelligence Approach*. Palo Alto, CA: Springer Science and Business Media; 2013.

How to cite this article: Santos R, Souza D, Santo W, Ribeiro A, Moreno E. Machine learning algorithms to detect DDoS attacks in SDN. *Concurrency Computat Pract Exper*. 2019;e5402. https://doi.org/10.1002/cpe.5402