Journal of Network and Systems Management (2023) 31:55
https://doi.org/10.1007/s10922-023-09741-4

RESEARCH

SDNTruth: Innovative DDoS Detection Scheme for Software-Defined Networks (SDN)

Tiago Linhares[1] · Ahmed Patel[1] · Ana Luiza Barros[1] · Marcial Fernandez[1]

Received: 31 October 2022 / Accepted: 12 May 2023 / Published online: 17 June 2023

Abstract

Software-Defined Networks (SDN) are a trending technology in the modern Internet by splitting control and data planes and using a central controller. An SDN controller provides flexible flow management at wire-speed packet forwarding in the Internet. The centralized control allows to implement detection and mitigation of security attacks inside the SDN controller. Typically, Distributed Denial of Service (DDoS) attacks pose an immense threat to Internet security. However, the prediction and prevention of DDoS attacks in SDN environments are a huge challenge. In this paper, we introduce a mechanism to mitigate DDoS attacks in SDN using statistical analysis and traffic entropy. To validate the proposal, a prototype was built in the Mininet tool. The accuracy and training time were compared against different Machine Learning algorithms. Finally, we expound on the effectiveness and limitation of the proposed solution as well as show our research plans and further research opportunities.

Keywords Software Defined Networks (SDN) · Network Functions Virtualization (NFV) · Distributed Denial of Service (DDoS) · Thread mitigation · Entropy

✉ Tiago Linhares
tiago.linhares@aluno.uece.br

Ahmed Patel
zapazap@gmail.com

Ana Luiza Barros
analuiza.barros@uece.br

Marcial Fernandez
marcial.fernandez@uece.br

[1] Computer Science Program, Universidade Estadual do Ceará (UECE), Av Dr Silas Munguba 1700, Fortaleza, CE 60.714-903, Brazil

1 Introduction

Software-Defined Network (SDN) is a network architecture approach that enables the network to be intelligently and centrally controlled by system programmable means that are formulated using a series of software applications using open Application Programming Interfaces (APIs). It is a network architecture that is dynamic, adaptable, manageable, and low-cost which assists operators to manage the entire network and its devices consistently and holistically, regardless of the underlying complexity of the underlying network technologies. On the other hand, Network Functions Virtualization (NFV) is the replacement of network appliance hardware with a Virtual Machine (VM) on top of virtualized infrastructure. The VMs use a hypervisor to run networking software and processes such as routing, gatewaying, and load balancing to provide systematic centralized network-wide infrastructure controls. The basic distinction between NFV and SDN is that NFV provides the basic networking functions and SDN undertakes higher-level management responsibilities to orchestrate overall network operations.

SDN networks expand the possibilities of network design and management, as they are based on the separation of the packet forwarding plane from the control plane. In this way, the network control is centralized in a software component called the *controller*. Centralized control simplifies the implementation of new services, the adoption of management policies, and the network reconfiguration requirements by software, since it becomes a programmable network [27]. There are some protocols to implement SDN networks. The most common is OpenFlow [28].

Denial of Services (DoS) are attacks where the network is flooded by many spoofed packets sent from compromised machines. One class of such attacks is the Distributed DoSs (DDoSs), when multiple compromised machines attack a target simultaneously [45]. This is further manifested by botnets used for many malicious purposes, including stealing personal data and passwords, ransomware, attacking public and private networks, exploiting users' computing power and Internet access, and a more serious one, executing DDoS attacks on a large scale. In short, botnets are a complex and continuously growing problem that poses a threat to both system and user confidence in the Internet [41].

DDoS attacks pose a serious threat to any network, including those based on SDN [37]. This type of attack aims quickly to deplete a target's communication and computational power by flooding it with a large volume of malicious traffic [16]. Detecting DDoS attacks is a complicated task, as attack packets can be mistaken for legitimate packets. Another difficulty is the large number of packets to be analyzed, which affects detection accuracy and fast response time to attacks [5]. The choice of an ideal classification technique for identifying DoS and DDoS attacks is extensively analyzed by Santos et al. [47].

DoS/DDoS attacks are realized in SDNs by creating several deceptive flows that flood the bandwidth of the data plane, control plane, the OpenFlow switches, and the SDN controller, which results in network failure for reaching legitimate hosts as well as the entire network [13].

Yan et al. [52] demonstrated how the SDN architecture can facilitate the detection and prevention of DDoS attacks in Cloud Computing and other SDN-based infrastructures such as IoT and critical utility serving infrastructures, which cannot afford to have major disruptions and downtimes. The authors demonstrate that some SDN networks characteristics, such as centralized control, software-based traffic analysis, global network view, and dynamic forwarding rules updating can be exploited to prevent DDoS attacks. They also pointed out in their paper, SDN networks still need more studies on vulnerabilities and threats detection with proper risk analysis.

With the growing adoption of SDN networks, it becomes necessary to implement DDoS attack mitigation systems in these computer networks. Two important characteristics of SDN networks can be exploited effectively to facilitate the detection of DDoS attacks: (1) the ease of use of management data analysis by the centralized SDN controller from the entire network in real-time, and (2) the programmability of the network devices that allows direct actions to mitigate the effects of the attack in a very short time. The idea of using flow abstraction to identify DoS attacks in computer networks is already successfully implemented in some real-life Intrusion Detection System (IDS) systems, as shown by Kakihata et al. [19].

This article proposes the *SDNTruth* system to detect and mitigate the effects of DDoS attacks on the data plane in SDN networks. The name *SDNTruth* refers to improving reliability and veracity in SDN networks by implementing security mechanisms against DDoS attacks. Thus, a flow analysis module was developed in an SDN controller that uses entropy evaluation and Z-Test to identify the anomalous behavior of a network flow. On every 30 entropy measurements, the average and standard deviation are calculated, and then, the Z-Test is used to assess the flow behavior, whether it is normal or not.

As will be detailed in Sect. 3, on every 30 entropy measurements, the average and standard deviation are calculated and, finally, the Z-Test is used to assess the flow behavior, whether it is normal or not. In this paper, we compare the *SDNTruth* results against traditional Machine Learning (ML) algorithms, such as Decision Tree, Random Forest (RF), Neural Network, Logistic Regression, Support Vector Machines (SVM), and Naive Bayes. This comparison aims to assess whether the results of our proposal are comparable to the results obtained by some traditional ML algorithms used in numerous research works in this field.

The contributions of this work are: (1) it proposes the *SDNTruth* system, based on entropy calculation and Z-test to detect and mitigate DDoS attacks on SDN networks, (2) the *SDNTruth* is compared to traditional ML algorithms, and, (3) it is demonstrated that *SDNTruth* gives similar detection accuracy, in shorter times and using fewer computational resources, compared to traditional ML algorithms.

This article is organized as follows: Sect. 2 presents some pertinent related works that guided this research. The proposed *SDNTruth* system architecture is presented in Sect. 3. In Sect. 4, the prototype is evaluated and in Sect. 5, the results are presented. Finally, Sect. 6 gives a concise discussion and in Sect. 7 the conclusions are presented and some important future research and development works are suggested to enhance the *SDNTruth* system and overcome its limitations.

2 Related Works

Oshima et al. [40] demonstrated that Z-Test had better false-negative results in detecting DoS/DDoS attacks. Using the Z-Test, it was possible to detect an attack with a smaller sample, allowing the identification of the attack in shorter time.

In David and Thomas [10], a technique for detecting DDoS attacks is proposed by calculating the entropy variation of the packet flow in traditional networks. An adaptive algorithm is used to calculate the entropy variation threshold, since network metrics and user behavior vary over time. The authors show that entropy measurement brings several benefits to the detection of DDoS attacks.

Mousavi and St-Hilaire [30] demonstrate how a DDoS attack can quickly exhaust the resources of an SDN controller. They propose a technique to detect an attack based on the packet destination IP address entropy calculation. The proposed technique permits detecting a DDoS before the first 500 malicious packets are received. This allows detection of the attack before the SDN controller is flooded with a huge number of Packet-In messages. In the controller, a function was added to create a *hash* table with IP packet information. If the packet is new, it is added to the table, but if the package is repeated, the packet number is incremented. In this work, entropy is calculated for every 30 packets, and by experimentation, a threshold was defined for the entropy value of each flow. If the entropy value is below this threshold, the traffic is considered benign; otherwise, an attack is identified.

Cui et al. [9] show a comprehensive review of the DDoS detection mechanisms for SDN using ML-based and threshold-based detection mechanisms. Singh and Behal [48] present a comprehensive review, research challenges, and future directions on the detection and mitigation of DDoS attacks in SDN. Dong et al. [12] present the state of the art of DDoS attacks in cloud computing scenarios using SDN. An important work about comprehensive security risk analysis in critical infrastructure environment is given by Qassim et al. [44]. Their work shows a survey of several techniques for detecting DDoS attacks in the Cloud Computing environment.

Tan et al. [49] propose a framework for the detection and defense of DDoS attacks in SDNs. The work deals with cooperative detection methods of the control plane and data plane. A DDoS detection trigger mechanism is deployed on the data plane, and when it detects some suspicious traffic, the controller is triggered. The controller detection mechanism is based on ML techniques combining K-Means and KNN algorithms to exploit flow characteristics. Rate and asymmetry are used to detect suspicious flows. And, depending on the findings, the controller executes defense actions. Compared to *SDNTruth*, the work uses ML to detect DDoS attacks on SDNs. However, it combines supervised and non-supervised algorithms, and the attack evaluation metric differing from the entropy.

Perez-Diaz et al. [43] presented an architecture for the identification and mitigation of Low-Rate DDoS (LR-DDoS) attacks in SDNs. According to the authors, this kind of attack is difficult to detect, particularly in an SDN. An advantage of the proposed architecture is to be modular, making it possible to easily replace

or enhance a module, or ML model. This flexibility is important when we are interested in discovering the best module configuration for different problems or different attacks. The IDS is trained with six ML models [i.e., J48, Random Tree, REP Tree, RF, Multi-Layer Perceptron (MLP), and SVMs)]. And because of the flexibility of the architecture and efficiency of the system, some modules, such as IDS, can be implemented in a separate hardware component outside the controller. In this way, the controller does not have to handle flow classification complexity, remaining simple to process incoming flows and make rapid decisions about those flows. The work uses some of the ML algorithms used in *SDNTruth*. One of the strengths of the work is the architecture's flexibility. Moreover, it addresses LR-DDoS, which *SDNTruth* proposes to deal with in future work, using a two-phase mechanism.

Yungaicela-Naula et al. [54] also present a modular and flexible architecture to detect DDoS attacks on an SDN-based environment. The solution considers different DoS/DDoS attacks, including both the transport and the application layers. For flow classification, it uses ML and Deep Learning (DL) models. The goal is to explore different ML/DL methods to determine which methods perform better detection under different types of attacks and conditions. ML-evaluated methods are SVM, RF, and K-Nearest Neighbor (KNN). For DL, the used methods are MLP, Convolutional Neural Network (CNN), Gated Recurrent Units (GRU), and Long Short-Term Memory (LSTM) neural network. Different from *SDNTruth*, in addition to ML algorithms the work also addresses DL. This improves the possibility of combinations to find out good learning models. One of the advantages of *SDNTruth*, on the other hand, is to provide a simple model that achieves good performance without increasing time consumption.

Variations in entropy are the method used by Mishra et al. [29] as a defensive mechanism for DDoS attacks. Despite the use of entropy, such as *SDNTruth*, they also develop a mitigation technique to reduce the severity of an attack. And our focus in this work is on detection solutions. The detection process is executed step by step, based on three thresholds, which are incoming packet flow, entropy, and count threshold. Unusual traffic is detected by comparing the flow rate with the incoming packet flow threshold. If the value is exceeded, statistical information is collected for entropy calculation. The value is then compared to the entropy threshold, similar to *SDNTruth*. After the attack happens, the counter value is incremented and those packets are marked. Then, for those specific packets that have the higher count value in comparison with the count threshold, alert messages of attack are generated. Then, the algorithm invokes the mitigation function. This recent work supports the idea of using entropy as a detection mechanism for DDoS attacks.

Yu et al. [53] propose a cooperative DDOS attack detection framework based on entropy and ensemble training in SDN, which uses the computing power of the edge switch to offload part of the detection tasks from the control plane to the data plane. The architecture evaluates the entropy on edge nodes and trains the RF model in ensemble learning, in which the output results are determined according to the score formed by the number of decision tree votes. The prototype was effective against SYN and ICMP Flood attacks.

Liu et al. [26] divide the attack detection method in the SDN environment into two levels of granularity. At the coarse granularity level, the attack source is located, reducing the detection scope. At this point, suspicious components and ports are detected by the information entropy detection mechanism. Then the fine-grained level, based on DL, distinguishes normal traffic from suspicious traffic through a packet-based detection mechanism using a CNN model. One interesting aspect of the work is the use it does of traffic information as images to present them to CNN. Each byte of the packet is converted into an unsigned number ranging from 0 to 255, converting the data packet into grayscale images. This very recently published work also uses entropy to detect attacks, such as *SDNTruth*, supporting our choice. Besides, the idea of bringing together entropy and traffic representation as images is a new way to profit from the DL CNN model.

Our unique innovative proposal combines the idea of parameters entropy variation of network flows, similar to David and Thomas [10] and Mousavi and St-Hilaire [30] works, and the Z-Test treating based attack detection, similar to the Oshima et al. [40] work. Compared to Yu et al. [53] and Yungaicela-Naula et al. [54] works, our solution can protect against more different DDoS attacks with better outcomes. Table 1 summarizes the main characteristics of related works.

3 SDNTruth: DDoS Attack Detection and Mitigation in SDN

The *SDNTruth*, DoS/DDoS Attack Detection System proposed in this paper, is hosted on the SDN centralized architecture to get network information to resolve security issues, DDoS being one important of them. *SDNTruth* takes advantage of the network global information got by a centralized SDN controller, allowing identify DDoS attacks originating from multiple sources entering into multiple network interfaces. However, we recognize the limitations of a centralized architecture in case of failures or outages and we propose some future works to overcome this issue. The reason for taking this approach is that the work of Mousavi and St-Hilaire [30] shows that this way is easier to get data from the networks when compared to

Table 1 Summary of related works

Reference	Entropy	Z-Test	ML	DL
[40]		•		
[10]	•			
[30]	•			
[49]		•		
[43]			•	
[54]			•	•
[29]	•			
[53]	•		•	
[26]				•
This work	•	•	•	

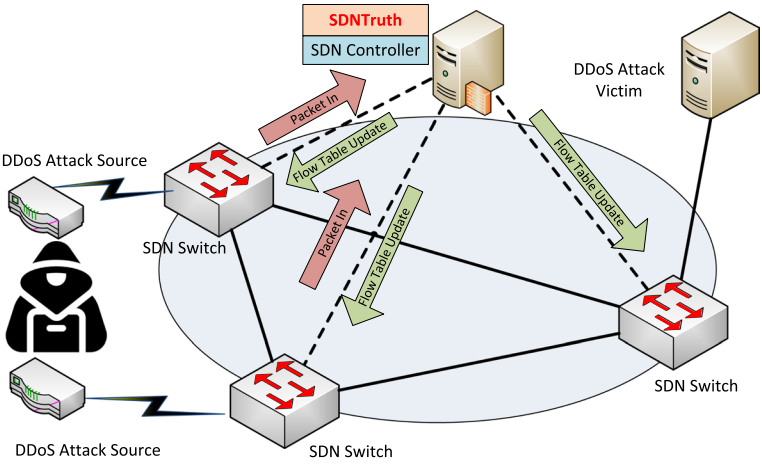


Fig. 1 SDNTruth in SDN architecture

traditional networks, i.e., using distributed algorithms and procedures, like Simple Network Management Protocol (SNMP) or *sFlow* protocols.

Figure 1 shows the *SDNTruth* in an SDN architecture, using commodity SDN switches and the SDN Controller. A DDoS attack can be performed outside or inside the network and the victim is inside the provider's network. The SDN controller calculates the flow entropy after one Packet-In message is received, and then, the entropy is calculated by reading the SDN statistics field of this flow. In our proposal, we assume that Packet-In messages are forwarded through the control plane. The data plane is susceptible to management control because intrusions can easily sip in by bypassing the control plan, which is detrimental to the activities of not only the data plane but the whole SDN system. After identifying the attack, a flow table update message is sent to all switches to block the attack. It is important to note that, as the monitor centralizes Packet-In message arrival information on the network, the analysis is performed globally, regardless of the number and the location of *bots* acting at a moment.

Figure 2 shows the components and flowchart of the *SDNTruth* System, which is detailed below.

Algorithm 1 shows the sequence of actions to detect a DDoS attack.

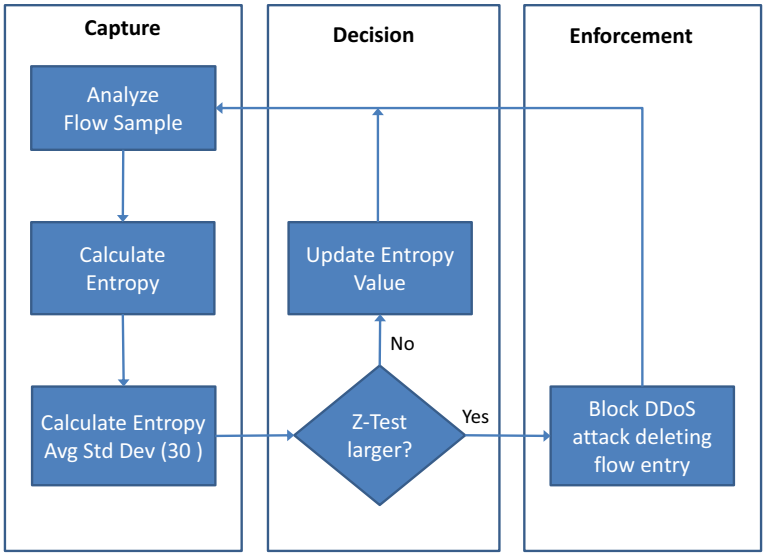


Fig. 2 SDNTruth flowchart diagram

Algorithm 1 SDNTruth: DoS/DDoS attack detection

```
Collects number of flows defined by the Admin
Calculates Initial Entropy
while Analyzing flow do
    for all 30 Entropy Samples do
        Calculates Average/Standard Deviation
        Calculates Z-Test against normal Entropy
        if Z-Test shows discrepancy then
            Discards attacker traffic
        else
            Updates normal Entropy value
        end if
    end for
end while
```

Before starting the protection, it is necessary to learn how to calculate the initial value of entropy. For each average and standard deviation measurement, it is verified whether there is a discrepancy between the observed value and the normal value. To identify the abnormal behavior, the following analysis is done: (1) entropy value variation is observed, and (2) Z-Test checks if the calculated entropy value is consistent with the entropy calculated in the previous time slot. Each step of this procedure is detailed below.

3.1 Capture Component

This component analyzes the flows and stores the flow parameters in the database:

- Source IP,
- Destination IP,
- Source Port,
- Destination Port.

Entropy is a measure of unpredictable information. Formally, given $X = \{x_1, x_2, \ldots, x_n\}$ a set of n distinct elements and considering $Y = \{y_1, y_2, \ldots, y_s\}$ a set of occurrences in the elements of X. For every x_i that appears in the set Y a number n_i of times then the probability of the occurrence of x_i in Y is $p(x_i) = \frac{n_i}{s}$. Entropy is defined in terms of probability as shown in (1):

$$H(Y) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i), \tag{1}$$

where n is the number of elements in Y. The maximum entropy value is got when each element in Y appears only once [15].

The first step of the methodology consists in collecting information from packet flow statistics. The statistics are obtained from OpenFlow multipart messages, OFPMP_FLOW_STATS, and OFPMP_{A}GGREGATE_STATS, since these are readily available. This information is extracted from each new stream transmitted and is stored for network behavior analysis. For each flow set, the entropy variation is calculated for each parameter. The entropy average value and standard deviation are calculated for every 30 samples collected, considering that the entropy average obeys a Normal Distribution [18]. By the Central Limit Theorem (CLT) [4], this is satisfied when we have a large number of samples [40].

3.2 Decision Component

In this component, a test is performed to detect if there is evidence of an attack. The mean and standard deviation values are compared using the Z-Test against previously measured values under normal network conditions. Before the detection phase, the system analyzes network traffic under normal conditions and saves these values for later comparison in the detection phase. For an attack to be detected, entropy divergence in the parameters of packet flows is required.

Z-Test is a statistical test to determine whether two population means are different when the variances are known and the sample size is large. A Z-Test is a hypothesis test in which the z-statistic follows a normal distribution [23], and the sample size is greater than 30. The t-test was considered, but it was adopted Z-test based on the work of Oshima et al. [40], which calculates the Z-Test as given by (2):

$$Z = \frac{|\overline{H_N} - \overline{H_A}|}{\sqrt{\frac{\sigma_n^2}{n} + \frac{\sigma_r^2}{r}}}, \tag{2}$$

where $\overline{H_N}$ and $\overline{H_A}$ are the averages of the entropy values under normal and under attack conditions, respectively. The σ_n and σ_r are the values of standard deviations under normal and under attack conditions, respectively. The n and r values are the values of the sample sizes under normal and under attack conditions. For the test, the distributions with means $\overline{H_N}$ and $\overline{H_A}$ must obey a normal distribution.

In our implementation, we use the Z-Test with a statistical significance of $\alpha = 5\%$, following a suggestion from Oshima et al. [40], calibration that gives the best accuracy. For this significance, if $Z \geq 1.64$ (tabulated value) the attack is detected. We work with the hypothesis that if there is a divergence in the values calculated by the Z-Test, there is evidence of a DDoS attack. When there is no attack evidence, the system will archive the results and continue the traffic analysis. If there is evidence of an attack, a countermeasure is taken.

To evaluate the proposal, an attack detection system prototype was implemented in the *FloodLight* controller [14]. The prototype was evaluated in emulation mode on the *Mininet* platform [20].

To compare the effectiveness and limitation of the proposed solution, the accuracy and training time are compared to a set of traditional ML algorithms.

3.3 Enforcement Component

If an attack is detected, the system checks the destination IP address for the largest number of flows and ceases their forwarding. The flow blocking is done by the controller's rules. The Access Control List (ACL) module allows the controller to configure proactively a set of rules for SDN flows, without the need to monitor Packet-In messages sent by SDN *switches*. The controller modifies the switch's flow table entry so that all packets with the victim's destination IP and port are discarded. In our previous work [25], it was shown better accuracy when using source/destination port and destination address. Once this is done, the victim's IP address and port are saved in the information table for further identification purposes. This module can also notify the administrator to take direct remedial action.

4 SDNTruth: DDoS Attack Detection System Evaluation

To validate the proposed system, a prototype based on the emulation technique was built. Several emulations were performed to test DDoS attacks, where the accuracy was measured to compare the training time and accuracy against traditional ML algorithms.

The **benign** traffic, i.e., harmless or well-intentioned traffic, used to evaluate the *SDNTruth* system was obtained from the PCAP file available on the *SimpleWeb* site. This file was captured by the *TCPDump* tool from the University of Twente

(Netherlands) network which has about 2000 students [3]. The DDoS attacks used to test the proposed system were a combination of the top-9 Akamai attacks [35], which include UDP fragmentation, DNS, NTP, CLDAP, UDP, CHARGEN, SYN, SSDP, and ACK attacks. The PCAP files used to emulate DDoS attacks were created by Nagy [31–34].

4.1 Test Environment

The *Mininet* environment was used to build the test scenarios, it is a tool widely used by researchers and developers for emulating SDN networks. With *Mininet* it is possible to emulate networks with several *hosts* and *switches* in systems with few computational resources [21]. The attack detection module has been implemented for the *FloodLight* controller. *FloodLight* is a controller implemented in Java that offers a modular architecture, which facilitates the implementation of new services [14]. To generate normal traffic and DDoS attack in *Mininet*, the PCAP files were played using *TCPReplay* from the Wireshark tool [8].

4.2 Topology

Figure 3 shows the topology created in *Mininet* for the experiments. The topology indicates one *host* responsible for playing the benign traffic and four *hosts*

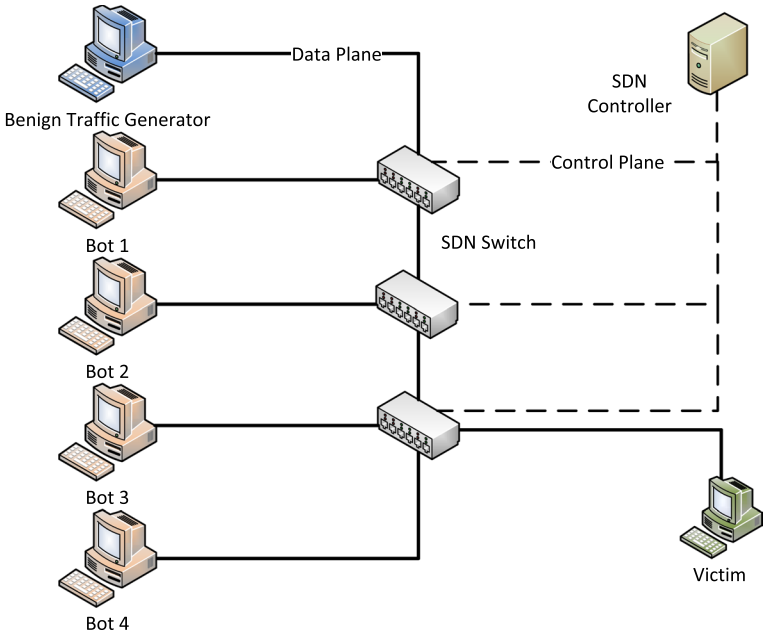


Fig. 3 Evaluation topology used in the tests

responsible for playing the malicious DDoS attacks, in addition to the SDN controller and the victim host.

4.3 Prototype Evaluation

To evaluate the *SDNTruth* System, the following procedure was performed. Initially, benign traffic PCAP files were reproduced in *Mininet* topology, as stated in the previous section. Benign traffic is played from benign traffic generator nodes representing normal Internet traffic. After calculating the initial entropy, this value is stored in the *SDNTruth* register.

After calculating the average and standard deviation of benign traffic, the DDoS attack traffic was started. PCAP files with different attacks are used to simulate a massive DDoS attack, one in each *bot* node. For the test scenarios, we compare normal network flows against flows with the DDoS attacks.

To compare the viability, effectiveness, and limitations of the proposed solution, we also evaluate for the same problem the accuracy and training time of traditional ML algorithms, such as Decision Tree, RF, Neural Network, Logistic Regression, SVM, and Naive Bayes. These ML algorithms were chosen by selecting the most relevant and up to date scientific papers available in published literature on the same topic [11, 17, 42, 50] and their availability in Apache Spark. All algorithms were calibrated for best accuracy, following the methodology presented in Sect. 4.4, to avoid distortions and biases in the comparison.

The ML evaluation was not run in the *Mininet* environment. The same PCAP files (benign and DDoS attack traffic) were converted to Comma-Separated Values (CSV) dataset using *tshark* software from the Wireshark tool [8]. To compare the results, a set of selected ML algorithms were executed in Apache Spark Platform [46] using 80% of the dataset for training and 20% for testing. Although we agree that the test environment differs from the environment used to test *SDNTruth*, it was useful to our main research question and goal, to compare the ML accuracy and training time to compare against *SDNTruth* to verify and validate it.

The PCAP files hosting benign and DDoS attack traffic are collected for further processing. *SDNTruth* used the *tshark* software from the Wireshark tool [8], resulting in converted CSV output files. These CSV files deliver a set of samples where each one of them is a packet, with packet-wise features. Packet-based classification has been explored before and criticized for its low effectiveness compared to flow-based classification [1].

Since *SDNTruth* is a flow-based classification approach, the statistics used to calculate entropies are retrieved from OpenFlow statistics of each flow. Literature analysis confirms that some flow-based DDOS attack datasets are used in ML applications that typically follow this method [2], that is the classifier collects the statistical features of the total flow, commencing from the sync packet to the end packet or until the network connection shuts down to obtain the complete flow representation. It typically aggregates packets that share the 5-tuple, i.e., the source port, destination port, source IP, destination IP, and the used protocol ID (i.e., TCP, UDP) [2].

Since the actual method deployed to evaluate the *SDNTruth* run ML on Apache Spark, converting all the string fields to numerical values was necessary. In the program script, the string was replaced with the hash64 in each field. Since the PCAP file already had IP and MAC addresses in string format, all fields were converted to a hash, and a new field, called FlowId was created, containing the hash of the string concatenating the 5-tuple fields. In hindsight, future work should look into using a flow extractor tool, e.g. Argus or CICFLowMeter [22], to process the PCAP file.

4.4 ML Algorithms Optimization

It is well known that finding the optimal model for an ML algorithm is a complicated and costly computational task. To reach a plausible solution, it is necessary to use techniques and strategies to find a satisfactory model without executing all ML algorithms with a series of hyperparameters [24].

In this experiment, to tune the hyperparameters of the selected ML algorithms, the *GridSearch* strategy was used. This method comprises testing ML algorithms with different hyperparameter values to find the best model that satisfies a metric. In addition, distributed computing could reduce computational time [36].

To correctly apply *GridSearch*, it is necessary to handpick the correct hyperparameters and their respective values to be tested. The only exception is the Naive Bayes algorithm, which will be explained later. In our work, the focus was to set the Stop Criteria parameters, such as *MaxDepth* (maximum tree depth) for tree-based algorithms and *MaxIter* (maximum number of iterations) for other algorithms.

The hyperparameter values selected to be tested should be chosen carefully. For tiny values, we get a non-satisfactory accuracy rate. For very large values, the computational cost would be high besides the risk of overfitting. Another important point is that the algorithms tested in this work belong to the Apache Spark framework, i.e., all the parameters were set according to the allowed values on this platform.

The hyperparameter's search space was defined as considering: (1) a small value, (2) an intermediate value, and (3) a large value. In the *GridSearch* strategy, each algorithm is executed three times, where each execution is assigned a small, a medium, and a large value until reaches the defined Stop Criteria.

As tree algorithms share the same hyperparameters, their configuration is quite similar. The only adjusted value is the MaxDepth. Other Decision Tree hyperparameter values are shown in Table 2, and for RF in Table 3. Hyperparameters not shown on these tables have been set to default Apache Spark values.

Table 2 Decision tree hyperparameters

Hyperparameter	Value
maxDepth	[1, 15, 30]
maxBin	64
impurity	Gini
cacheNodeIds	False
maxMemoryInMB	1024

Table 3 Random Forest hyperparameters

Hyperparameter	Value
maxDepth	[1, 15, 30]
maxBin	64
impurity	Gini
cacheNodeIds	False
maxMemoryInMB	1024
numTrees	50
featureSubsetStrategy	Auto
checkpointInterval	10
Seed	None
checkpointInterval	1.0

The SVM and MLP algorithms share the same hyperparameter (MaxIter) that was adjusted by the *GridSearch* method. MaxIter was adjusted to find the desired accuracy value without overfitting the model. The MaxIter values used were 10, 50, and 100.

On the SVM algorithm, the Linear Kernel was chosen and other hyperparameters were set to the Apache Spark default values, shown in Table 4. For the MLP algorithm, we set 10 intermediate layers and two output classes, benign or DDoS. Other parameters are shown in Table 5. Finally, Logistic Regression used most of the default values, varying only the MaxIter. The parameter values are shown in Table 6.

To optimize the Naive Bayes algorithm, it was necessary to use another approach. As Naive Bayes is a statistical algorithm and does not have stopping criteria, the

Table 4 SVM hyperparameters

Hyperparameter	Value
maxIter	[10, 50, 100]
regParam	0.1
standardization	True
fitIntercept	True
aggregationDepth	2
tol	1e−6
kernel	Linear

Table 5 MLP hyperparameters

Hyperparameter	Value
maxIter	[10, 50, 100]
Seed	1234
blockSize	128
tol	1e−6
Layers	[26, 10, 2]

Table 6 Logistic regression hyperparameters

Hyperparameter	Value
maxIter	[10, 50, 100]
fitIntercept	True
aggregationDepth	2
tol	1e−6
threshold	0.5

only tunable hyperparameter is the smoothing one. *Additive Smoothing* is the technique used to smooth categorical data, i.e., the smoothing process creates an approximation function that tries to capture the most important patterns in the dataset [51]. The values used to test this algorithm were selected randomly. The *Random GridSearch* method was used, where a set of random values is selected randomly, and using accuracy a criterion for choosing the best model.

5 Results

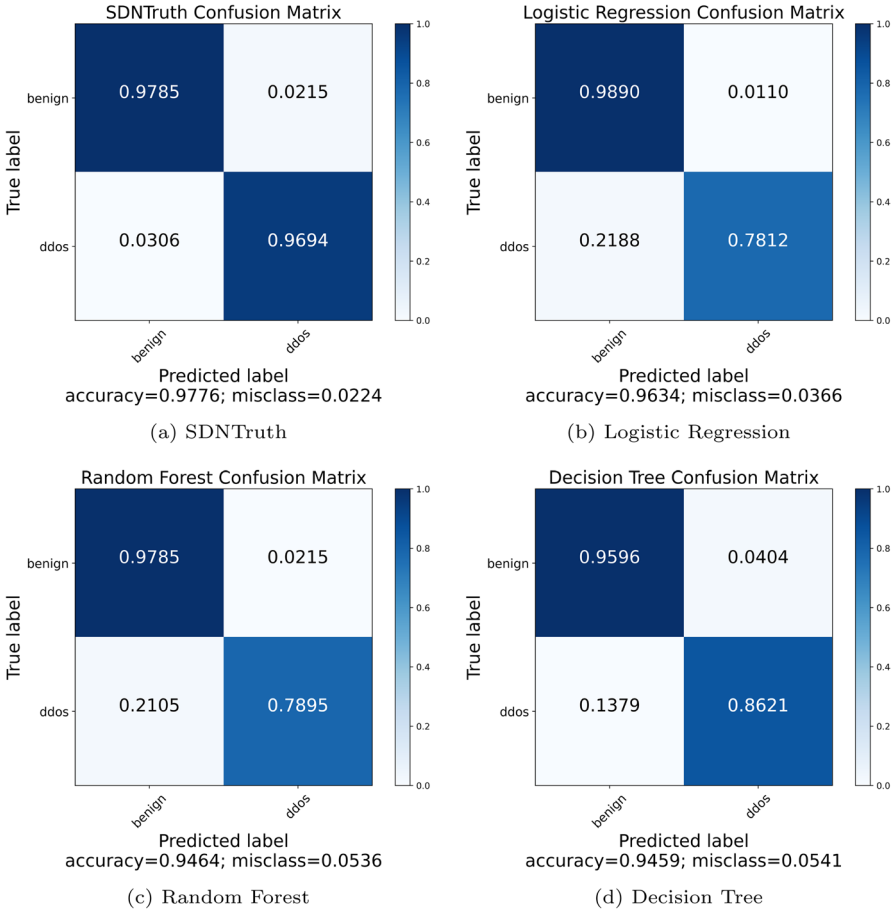
In this section, we present the results from the experiments. The *SDNTruth* considers the use of 3 parameters, source IP, destination IP, and destination Port; and 3000 flows "training exercises", which gives good accuracy of detection, as shown in Lima and Fernandez [25]. The increase in the number of analyzed flows causes a small impact on accuracy and increases the "training" time. We also noticed an increase in False Negatives when evaluating four parameters (including destination port), regardless of the number of flows inspected. This is because by increasing the number of parameters evaluated, the flows become more different, reducing the entropy, which identifies a false DDoS attack.

Our results are better than the ones provided by Mousavi and St-Hilaire [30], which used a similar method but needs 4000 flows to get 96% accuracy. The scientific justification for our better result is the use of the Z-Test, which gives more accurate answers with smaller samples, i.e., it is more sensitive to detect network traffic variation.

Figure 4 shows the confusion matrix of *SDNTruth*, Logistic Regression, Decision Tree, and RF evaluation. These algorithms were selected because they give better accuracy in a reasonable time compared to other algorithms. The *SDNTruth* presents good accuracy and very few False Negative events, as it is expected in a DDoS defense system.

On the one hand, ML algorithms offer good accuracy results. Although, most of them give a higher value of False Negatives, which is bad for security. because it allows DDoS attacks undetected. *SDNTruth*, on the other hand, acts more strongly to identify attacking flows, reducing False Negatives and maintaining low False Positives.

Table 7 shows the accuracy and training time for *SDNTruth* and selected ML algorithms, such as Decision Tree, RF, Neural Network, Logistic Regression, SVM,



(a) SDNTruth (b) Logistic Regression

(c) Random Forest (d) Decision Tree

Fig. 4 Confusion matrix: SDNTruth, Logistic Regression, Random Forest, and Decision Tree

Table 7 SDNTruth vs. ML algorithms summary

Algorithm	Accuracy	Training time (s)
SDNTruth	0.977	0.3[a]
Logistic Regression	0.963	855
Random Forest	0.946	174
Decision Tree	0.945	56
Support Vector Machines	0.818	6226
Neural Network	0.785	890[b]
Naive Bayes	0.644	54

[a]Considering the time to calculate the first entropy value, 3000 packets on a 1 Gbps link

[b]Time calculated using a GPU board

and Naive Bayes. All the evaluations were performed on a bi-processor Lenovo SR630 server. An important observation is that the MLP algorithm performance was measured with a GPU board because the testing time on the server was taking extremely long.

Note that the best training time was got for Logistic Regression algorithm. However, its training time is much longer compared to the training time of *SDNTruth*, giving rough accuracy. Overall, our *SDNTruth* is superior because it provides good security level in SDN networks with low use of computational resources.

6 Discussions

As shown in our series of experiments, *SDNTruth* demonstrates effectiveness in detecting DDOS attacks with high accuracy in a very short time. The prototype was tested against different DDoS attacks running simultaneously, which is a real scenario that causes difficulties in identifying attacks using traditional tools. Since the experiments were performed in emulation, we can believe that it will also be effective in a real-world SDN network.

We can speculate on some limitations of *SDNTruth*. The first limitation refers to the architecture's scalability since it makes the decision centrally. However, since the entropy and Z-test metric calculation is quite simple, we believe it will not significantly impact the controller's performance. Surely, the normal packet forwarding decision in the controller must be more powerful. In any way, an in-depth study on *SDNTruth* scalability in a real large-scale environment will be interesting.

Another limitation is the possibility that a new subtle attack can throw *SDNTruth* off. It is possible to imagine that a new DDoS attack presents smooth changes over time, causing *SDNTruth* to perceive it as a normal traffic variation and not identify the attack. To handle slow-rate DDoS attacks, we propose a solution in two phases. First, the algorithm detects traffic characteristics. If it is a normal flow, *SDNTruth* runs as expected. If there is a flow with a few packets (i.e., less than 30 packets), it will be sent to an offline pre-trained ML model implementation, like logistic regression or decision tree, to check the traffic rationality. The model can be retrained offline and adjusted to real traffic variation, detecting slow-rate DDoS attacks.

Our experiments considered only one victim equipment of the attacks. This is a clear limitation in a real cloud provider environment where multiple users may suffer different attacks from different sources. Perhaps analyzing customer network segments within the cloud infrastructure independently could solve this problem. An evolutionary ML system combining various attack profiles might be interested.

As stated before, we believe *SDNTruth* can be used in real production networking and services environments. However, extensive testing to validate the proposal is necessary.

Although beyond the scope of this paper, it will also be worth investigating *SDNTruth* in a distributed SDN controller architecture setting and then comparing it against the central SDN controller architecture approach to determine system complexities, efficiency, and effectiveness.

Finally, it is obvious and worth mentioning, that system security is never foolproof and can always be cracked in multiple ways like an attack deceptively generated from trusted IP sources can send malicious network traffic which the *SDNTruth* system cannot predict with 100% certainty. Therefore, it would be necessary for future operational *SDNTruth*-type solutions for practical network infrastructures to have multiple switches and a distributed set of autonomous managed controllers coordinating jointly in the network to invoke comprehensive network packet analyzers. The four features used in the *SDNTruth* statistical analysis and entropy phases can be more comprehensively extracted and used with new advances in ML algorithms and computational intelligence scenarios to have a better accurate and clever prediction of DDoS and other malicious security breaching traffic.

To address distribution, one solution could be a multi-agent system with intelligent agents reasoning and acting in a distributed way to detect and prevent attacks. Each node and its agent can exchange information and collaborate on a global solution. We can work with a reinforcement environment where agents can learn from their previous experiences to improve their behavior. We can also think about multi-objective functions considering different important metrics for traffic analysis and attack detection, to optimize all of them together.

7 Conclusions and Future Works

SDNs have emerged as one of the main methods to implement programmable networks by pure software. It offers the necessary means to quickly adapt networks to adapt networks quickly to the changes required by current Cloud Computing and IoT infrastructures and their supporting applications. However, techniques to guarantee the security of SDN networks against DoS and DDoS attacks are still a big problem, both theoretically and practically. Research and development continue to be carried out.

This article presented the *SDNTruth* system, an architecture to detect DDoS attacks in SDN networks using statistical techniques with low computational demand and execution time. The proposed *SDNTruth* system leverages the information collected by the centralized SDN controller to analyze packet flows for security breaches and intrusions, specifically DDoS attacks. The information is analyzed using entropy and statistical Z-test to detect any anomaly in a DDoS attack traffic. Once the detection is made, the attack can be mitigated by interrupting the SDN controller flow of packets destined for the victim.

To evaluate the proposed *SDNTruth* system, a prototype was implemented in the *FloodLight* controller and tested in the *Mininet* environment. The prototype could detect an attack by analyzing the set of flows entering the network and blocking malicious packets to the victim. In the experiments conducted, good performance was observed in the detection of attacks with accuracy above 97% after analyzing 3000 packets. When we increased the number of flows for detection, the *SDNTruth* achieved an accuracy of 100%, the proposed solution, SDNTruth, is compared against usual ML algorithms, demonstrating good accuracy and low training time.

We argued that *SDNTruth* System could become essential for securing the next generation of SDN infrastructure, providing fewer security vulnerabilities and risks, particularly DDoS attacks. The most important criteria were to demonstrate that statistical techniques can produce good results in detecting DDoS attacks in SDN with much less consumption of computational resources.

In future work, we suggest investigating new approaches to reduce the time for identifying DDoS attacks. This enhancement should also improve the *SDNTruth* accuracy to reach the best ML algorithm accuracy. It is necessary to study the impact of other flow attributes to improve DDoS attack identification, e.g., packet type, packet size, flow time, and others.

The *SDNTruth* system should be applied and tested (experimented with) in real or pilot SDN-based networks to truly evaluate and prove their viability and effectiveness. An in-depth study on scalability in a real large-scale environment will be necessary. We do not believe that running part of the algorithm on the endpoints is a solution as proposed by Yu et al. [53]. One possibility is to consider distributed SDN controller architectures as indicated by Oktian et al. [39]. We can use distributed SDN controller architecture and use Commercial off-the-shelf (COTS) SDN switches and their standard functions, instead of Yu et al. [53] proposal to calculate the entropy on edge nodes to ease the central controller. Besides the scalability, the distributed controller approach provides a fault-tolerant architecture Obadia et al. [38], which would be the optimal solution for large complex network infrastructures.

Our mission and research objectives in the *SDNTruth* were to deal with classical ML models to improve DDoS attack detections. To address some points about the ML models, we propose in the next steps of our research work program to deal with DL models, more specifically with RNNs (Recurrent Neural Networks). Furthermore, we consider using different datasets from the one used in this paper and also improve the solution presented, developing a model for the multiclass classification of DDoS attacks, instead of implementing the binary one.

In this *SDNTruth* work, we dealt with classic ML models for improving DDoS attack detections. As future work, while outside the scope of this *SDNTruth* paper, we will consider DL and the development of DL solutions to process big volumes of data, such as traffic data generated in an SDN network. Different datasets could be used, such as CICDS2017 [6] and CIC-DDoS2019 [7].

To deal with the continuous nature of the network traffic, since packets are continuously generated, one of the most promising DL solutions is RNNs, like GRU and LSTM. Experiments involving these models will be conducted in two phases.

In the first phase, GRU and LSTM models will be developed. At this point, *Automated Machine Learning (AutoML)* will be used to improve the hyperparameters tuning and find out the best models since these are challenging and time demanding tasks in building DL solutions. The goal is not only to propose a model, but to try to find the best one, or near the best since the search is sometimes oriented by metaheuristics. The idea behind AutoML is to automate the ML pipeline, and at the first moment, we will deal with hyperparameter tuning and network architecture discovery steps.

In the second phase, DL ensembles will be involved. Ensembles execute collaboratively to reach better results, leading to models with better generalization and

better performance. They combine individual models, where each one contributes to the definition of the best general model. In our future experiments, ensembles will be defined considering the results and behavior of each model built in the first phase.

Also, one other possible idea that we are currently discussing within our research group is to consider not only the binary classification of attacks (DDoS or not DDoS-benign) but also all the types of DDoS attacks beyond what we know to date by extending the classification derived from new DL developed techniques and algorithms handling big data flows.

Acknowledgements Prof. Marcial Fernandez is sponsored by DT/CNPq/Brazil Grant Number 313231/2019-2.

Author Contributions TM evaluated and wrote the ML part. AP wrote the discussion. ALB wrote the ML part. MF wrote SDNTruth. All authors contributed and reviewed parts of the manuscript.

Funding This research is partially funded by Prof. Marcial Fernandez DT/CNPq/Brazil Grant Number 313231/2019-2.

Data Availability All datasets are public and freely available.

Declarations

Conflict of interest The authors declare that they have no competing interests.

Ethical Approval Not applicable.

References

1. Andreas, B., Dilruksha, J., McCandless, E.: Flow-based and packet-based intrusion detection using BLSTM. SMU Data Sci. Rev. **3**(3), 8 (2020)
2. Azab, A., Khasawneh, M., Alrabaee, S., et al.: Network traffic classification: techniques, datasets, and challenges. Digit. Commun. Netw. (2022). https://doi.org/10.1016/j.dcan.2022.09.009
3. Barbosa, R.R.R., Sadre, R., Pras, A., et al:. Simpleweb. University of Twente Traffic Traces Data Repository (2010)
4. Barron, A.R.: Entropy and the central limit theorem. Ann. Probab. **14**(1), 336–342 (1986)
5. Braga, R., Mota, E., Passito, A.: Lightweight DDoS flooding attack detection using NOX/OpenFlow. In: IEEE 35th Conference on Local Computer Networks (LCN2010), 2010, pp 408–415. IEEE (2010)
6. Canadian Institute for Cybersecurity: Intrusion Detection Evaluation Dataset (CIC-IDS2017). Canadian Institute for Cybersecurity (2017). https://www.unb.ca/cic/datasets/ids-2017.html. Accessed Dec 2022
7. Canadian Institute for Cybersecurity: DDoS Evaluation Dataset (CIC-DDoS2019). Canadian Institute for Cybersecurity (2019). https://www.unb.ca/cic/datasets/ddos-2019.html. Accessed Dec 2022
8. Combs, G.: Wireshark network protocol analyzer (2023). https://www.wireshark.org/. Accessed Jan 2023
9. Cui, Y., Qian, Q., Guo, C., et al.: Towards DDoS detection mechanisms in software-defined networking. J. Netw. Comput. Appl. **190**(103), 156 (2021). https://doi.org/10.1016/j.jnca.2021.103156
10. David, J., Thomas, C.: DDoS attack detection using fast entropy approach on flow-based network traffic. Procedia Comput. Sci. **50**, 30–36 (2015)
11. de Lima Filho, F.S., Silveira, F.A., de Medeiros Brito Junior, A., et al.: Smart detection: an online approach for DoS/DDoS attack detection using machine learning. Secur. Commun. Netw. **2019**, 1–15 (2019)

12. Dong, S., Abbas, K., Jain, R.: A survey on distributed denial of service (DDoS) attacks in SDN and cloud computing environments. IEEE Access **7**, 80813–80828 (2019). https://doi.org/10.1109/ACCESS.2019.2922196
13. Eliyan, L.F., Di Pietro, R.: DoS and DDoS attacks in software defined networks: a survey of existing solutions and research challenges. Future Gener. Comput. Syst. **122**, 149–171 (2021). https://doi.org/10.1016/j.future.2021.03.011
14. Erickson, D.: FloodLight Java based OpenFlow controller (2022). https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/overview. Accessed Jan 2022
15. Feinstein, L., Schnackenberg, D., Balupari, R., et al.: Statistical approaches to DDoS attack detection and response. In: Proceedings of DARPA Information Survivability Conference and Exposition, 2003, pp 303–314. IEEE (2003)
16. Gulisano, V., Callau-Zori, M., Fu, Z., et al.: STONE: a streaming DDoS defense framework. Expert Syst. Appl. **42**(24), 9620–9633 (2015)
17. He, Z., Zhang, T., Lee, R.B.: Machine learning based DDoS attack detection from source side in cloud. In: 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud), 2017, pp 114–120. IEEE (2017)
18. Jain, R.: Art of Computer Systems Performance Analysis: Techniques for Experimental Design Measurements Simulation and Modeling, 2nd edn. Wiley, Hoboken (2008)
19. Kakihata, E.M., Sapia, H.M., Oiakawa, R.T., et al.: Intrusion detection system based on flows using machine learning algorithms. IEEE Lat. Am. Trans. **15**(10), 1988–1993 (2017)
20. Lantz, B., Heller, B.: Mininet: rapid prototyping for Software Defined Networks (2021). http://mininet.org/. Accessed July 2021
21. Lantz, B., Heller, B., McKeown, N.: A network in a laptop: rapid prototyping for software-defined networks. In: Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, 2010, pp 19. ACM (2010)
22. Lashkari, A.H., Zang, Y., Owhuo, G., et al.: CICFlowMeter. GitHub[vid 2021-08-10] Dostupné z (2017). https://www.github.com/ahlashkari/CICFlowMeter/. Accessed Jan 2023
23. Lawley, D.: A generalization of Fisher's z test. Biometrika **30**(1/2), 180–187 (1938)
24. Li, L., Sparks, E., Jamieson, K., et al.: Exploiting reuse in pipeline-aware hyperparameter tuning (2019). arXiv preprint. http://arxiv.org/abs/1903.05176
25. Lima, N.A., Fernandez, M.P.: Towards an efficient DDoS detection scheme for software-defined networks. IEEE Lat. Am. Trans. **16**(8), 2296–2301 (2018). https://doi.org/10.1109/TLA.2018.8528249
26. Liu, Y., Zhi, T., Shen, M., et al.: Software-defined DDoS detection with information entropy analysis and optimized deep learning. Future Gener. Comput. Syst. **129**, 99–114 (2022)
27. Luo, S., Wu, J., Li, J., et al.: A defense mechanism for distributed denial of service attack in software-defined networks. In: Ninth International Conference on Frontier of Computer Science and Technology (FCST2015), 2015, pp 325–329. IEEE (2015)
28. McKeown, N., Anderson, T., Balakrishnan, H., et al.: OpenFlow: enabling innovation in campus networks. ACM SIGCOMM Comput. Commun. Rev. **38**(2), 69–74 (2008)
29. Mishra, A., Gupta, N., Gupta, B.: Defense mechanisms against DDoS attack based on entropy in SDN-cloud using pox controller. Telecommun. Syst. **77**(1), 47–62 (2021)
30. Mousavi, S.M., St-Hilaire, M.: Early detection of DDoS attacks against SDN controllers. In: International Conference on Computing, Networking and Communications (ICNC2015), 2015, pp 77–81. IEEE (2015)
31. Nagy, B.: CharGEN UDPfrag TCP SYN multivector DDoS attack. Zenodo (2021a). https://doi.org/10.5281/zenodo.5578700
32. Nagy, B.: CLDAP DNS multivector DDoS attack. Zenodo (2021b). https://doi.org/10.5281/zenodo.5572097
33. Nagy, B.: ICMP, UDP, TCP SYN multivector DDoS attack. Zenodo (2021c). https://doi.org/10.5281/zenodo.5578703
34. Nagy, B.: UDP flood attack sample. Zenodo (2021d). https://doi.org/10.5281/zenodo.5578712
35. Nagy, B., Orosz, P., Tóthfalusi, T., et al.: Detecting DDoS attacks within milliseconds by using FPGA-based hardware acceleration. In: NOMS 2018—2018 IEEE/IFIP Network Operations and Management Symposium, 2018, pp 1–4 (2018). https://doi.org/10.1109/NOMS.2018.8406299
36. Nguyen, N., Khan, M.M.H., Wang, K.: Towards automatic tuning of Apache Spark configuration. In: 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), 2018, pp. 417–425. IEEE (2018)

37. Nunes, B.A.A., Mendonca, M., Nguyen, X.N., et al.: A survey of software-defined networking: past, present, and future of programmable networks. IEEE Commun. Surv. Tutor. **16**(3), 1617–1634 (2014)
38. Obadia, M., Bouet, M., Leguay, J., et al.: Failover mechanisms for distributed SDN controllers. In: 2014 International Conference and Workshop on the Network of the Future (NOF), 2014, pp 1–6 (2014). https://doi.org/10.1109/NOF.2014.7119795
39. Oktian, Y.E., Lee, S., Lee, H., et al.: Distributed SDN controller system: a survey on design choice. Comput. Netw. **121**, 100–111 (2017). https://doi.org/10.1016/j.comnet.2017.04.038
40. Oshima, S., Nakashima, T., Sueyoshi, T.: Early DoS/DDoS detection method using short-term statistics. In: International Conference on Complex, Intelligent and Software Intensive Systems (CISIS2010), 2010, pp 168–173. IEEE (2010)
41. Patel, A., Taghavi, M., Bakhtiyari, K., et al.: An intrusion detection and prevention system in cloud computing: a systematic review. J. Netw. Comput. Appl. **36**(1), 25–41 (2013). https://doi.org/10.1016/j.jnca.2012.08.007,
42. Pei, J., Chen, Y., Ji, W.: A DDoS attack detection method based on machine learning. J. Phys. Conf. Ser. **1237**, 032040 (2019)
43. Perez-Diaz, J.A., Valdovinos, I.A., Choo, K.K.R., et al.: A flexible SDN-based architecture for identifying and mitigating low-rate DDoS attacks using machine learning. IEEE Access **8**, 155859–155872 (2020)
44. Qassim, Q.S., Jamil, N., Daud, M., et al.: A review of security assessment methodologies in industrial control systems. Inf. Comput. Secur. **27**(1), 47–61 (2019)
45. Razak, T.A., et al.: A study on IDS for preventing Denial of Service attack using outliers techniques. In: IEEE International Conference on Engineering and Technology (ICETECH2016), 2016, pp 768–775. IEEE (2016)
46. Salloum, S., Dautov, R., Chen, X., et al.: Big data analytics on Apache Spark. Int. J. Data Sci. Anal. **1**(3), 145–164 (2016)
47. Santos, K.R., Silva, I.R., Fagundes, R.A.A.: Classifiers comparison for attack detection in computer networks. IEEE Lat. Am. Trans. **15**(1), 87–96 (2017)
48. Singh, J., Behal, S.: Detection and mitigation of DDoS attacks in SDN: a comprehensive review, research challenges and future directions. Comput. Sci. Rev. **37**(100), 279 (2020). https://doi.org/10.1016/j.cosrev.2020.100279
49. Tan, L., Pan, Y., Wu, J., et al.: A new framework for DDoS attack detection and defense in SDN environment. IEEE Access **8**, 161908–161919 (2020)
50. Tuan, T.A., Long, H.V., Son, L.H., et al.: Performance evaluation of botnet DDoS attack detection using machine learning. Evol. Intell. **13**, 283–294 (2020)
51. Valcarce, D., Parapar, J., Barreiro, Á.: Additive smoothing for relevance-based language modelling of recommender systems. In: Proceedings of the 4th Spanish Conference on Information Retrieval, 2016, pp. 1–8 (2016)
52. Yan, Q., Yu, F.R., Gong, Q., et al.: Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing: a survey, some research issues, and challenges. IEEE Commun. Surv. Tutor. **18**(1), 602–622 (2016)
53. Yu, S., Zhang, J., Liu, J., et al.: A cooperative DDoS attack detection scheme based on entropy and ensemble learning in SDN. EURASIP J. Wirel. Commun. Netw. **1**, 90 (2021). https://doi.org/10.1186/s13638-021-01957-9,
54. Yungaicela-Naula, N.M., Vargas-Rosales, C., Perez-Diaz, J.A.: SDN-based architecture for transport and application layer DDoS attack detection by using machine and deep learning. IEEE Access **9**, 108495–108512 (2021)

Tiago Linhares is a graduate in Science Computing at the State University of Ceará (UECE) in 2019 and received his M.Sc. degree in Computer Science from the same university in 2022. Currently he is studying in the doctoral course at the same university. His interests are in Computational Intelligence, including Machine Learning, Artificial and Deep Neural Networks, parallelism, Hyperparameter optimization and distributed systems.

He has worked as a secondary-level educator in schools in the state, and is looking to improve himself to coordinate projects at the university. He works on several research papers, assisting other laboratories in various partnerships. He is a member of the digital systems laboratory (LASID), works as a researcher and mentor for other scientific works.

Ahmed Patel received his M.Sc. & Ph.D. degrees in Computer Science from Trinity College Dublin (TCD), University of Dublin in 1978 & 1984 respectively, specializing in the design, implementation & performance analysis of packet switched networks. He is Research Professor at Universidade Estadual do Ceará (UECE), Fortaleza, Brazil with key research interest in Advanced Computer Networking, Internet of Things, Cloud Computing, Big Data, Predictive Analysis, Use of Advanced Computing Techniques, Impact of e-social Networking, Closing the digital divide ICT gap and ICT Project Management. He has published well over 276 technical & scientific papers & co-authored three books, two on computer network security & the third on group communications. He co-edited one book on distributed search systems for the Internet and also co-edited & co-authored another book entitled: "Securing Information & Communication Systems: Principles, Technologies & Applications". He is a member of the Editorial Advisory Board of several International Journals and has participated in Irish, Malaysian, Brazilian and European funded research projects.

Ana Luiza Barros received her M.Sc. degree in Computer Science from the Federal University of Minas Gerais (UFMG) in 1996 and her Ph.D. degree in Teleinformatics Engineering from the Federal University of Ceará (UFC) in 2013. She is professor and researcher at the State University of Ceará (UECE), Fortaleza, Brazil. Her interests are in Computational Intelligence, including Machine Learning, Artificial and Deep Neural Networks, Pattern Recognition and Evolutionary Computation. She is member of the State Sectorial Chamber of the Productive Chain of ICT. She has worked on several research and development projects with universities and companies. Coordinates the Lab LAURA (Learn, Understand, Reason, Act) and is member of LAIS (Associated Innovation and Sustainability Laboratories), both at UECE.

Marcial Fernandez is a graduate of the Electronic Engineer Faculty at the Federal University of Rio de Janeiro in 1988 and received his M.Sc. and D.Sc. in Electric Engineering from the same university in 1998 and 2002 respectively. He completed his postdoctoral research in Future Internet at the Technische Universitat Berlin TU-Berlin in 2010. Prof. Fernandez is currently an Associate Professor at Universidade Estadual do Ceara UECE). His academic teaching and research are in the subject areas of Computer Communication Networks, Computer Architecture, Distributed Systems, Network Security, Parallel and Concurrent Programming. His research interests are in Software Defined Networks (SDN), Network Management, Network Provisioning, Content Distribution Networks, Datacenter Infrastructures, Internet of Things (IoT), Cloud Computing and Network Security. He has special expertise and skills in the application and use of Fuzzy Logic and its policies. He has published over 70 papers in scientific journals, conference proceedings and book chapters plus one book.