Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

An ecosystem for anomaly detection and mitigation in software-defined networking

Luiz Fernando Carvalho[a], Taufik Abrão[b], Leonardo de Souza Mendes[c], Mario Lemes Proença Jr.[a,*]

[a] Computer Science Department, State University of Londrina, Londrina, Paraná, 86057-970, Brazil
[b] Department of Electrical Engineering, State University of Londrina, Paraná 86057-970, Brazil
[c] Department of Communications (DECOM), School of Electrical and Computer Engineering, University of Campinas (UNICAMP), Campinas 13083-970, Brazil

A R T I C L E I N F O

Article history:
Received 24 November 2017
Revised 11 February 2018
Accepted 14 March 2018
Available online 15 March 2018

Keywords:
Anomaly detection
Software-defined networking (SDN)
OpenFlow
Multinomial logistic regression

A B S T R A C T

Along with the rapid growth of computer networks comes the need for automating management functions to prevent errors in decision-making and reduce the cost of ordinary operations. Software-defined networking (SDN) is an emergent paradigm that aims to support next-generation networks through its flexible and powerful management mechanisms. Although SDN provides greater control over traffic flow, its security and availability remain a challenge. The major contribution of this paper is to present an SDN-based ecosystem that monitors network traffic and proactively detects anomalies which may impair proper network functioning. When an anomalous event is recognized, the proposal conducts a more active analysis to inspect irregularities at the network traffic flow level. Detecting such problems quickly is essential to take appropriate countermeasures. In this manner, the potential for centralized network monitoring based on SDN with OpenFlow is addressed in order to evaluate mitigation policies against threats. Experimental results demonstrate the proposed ecosystem succeeds in achieving higher detection rates compared to other approaches. In addition, the performance analysis shows that our approach can efficiently contribute to the network's resilience.

1. Introduction

The rigid structure of traditional networks increases and complicates the task of managing them as they expand. As demand for real-time applications increases, it has become difficult to scale existing networks while ensuring availability and security without degradation of performance. To express the policies of agreed service levels (ASL) and to maintain quality of service (QoS), network operators need to configure each device separately as a heterogeneous collection of switches, routers, middleboxes, and so on, using low-level commands specific to each equipment supplier (Kreutz, Ramos, & Verissimo, 2013). Changes in such policies are time-consuming and usually require a large staff to focus efforts on making thousands of changes manually in network components. In this scenario, despite frequent investments in communications innovation, dissatisfaction with the capacity of traditional networks to adapt to the changes required by new technologies will become increasingly evident (Jammal, Singh, Shami, Asal, & Li, 2014).

Software-defined networks (SDNs) have emerged as the next-generation networking paradigm aiming to improve network resource utilization, simplify network management, reduce operating costs, and promote evolution (Lin, Wang, & Luo, 2016). To achieve these benefits, SDN separates the data plane from the control plane by removing decisions from the forwarding devices (e.g., routers and switches), enabling forwarding hardware programming through a standardized interface (commonly using the OpenFlow protocol). A controller guides the control plane using applications to define the behavior of the network-forwarding infrastructure. With the separation of planes, various network devices can share the same controller. A significant advantage is that if there is the need for policy changes or the establishment of service quality, it can be accomplished through control plane programming, which uses a more intuitive high-level language than device configuration commands. Indeed, the administrator can modify network forwarding rules, prioritize, or even block specific types of traffic.

Although SDN networks provide greater control over traffic flow, owing to their dynamic nature, they have also introduced new challenges and issues to be addressed. Researchers argue that

* Corresponding author.
E-mail addresses: luizfcarvalhoo@gmail.com (L.F. Carvalho), taufik@uel.br (T. Abrão), lmendes@decom.fee.unicamp.br (L.d.S. Mendes), proenca@uel.br (M.L. Proença Jr.).

https://doi.org/10.1016/j.eswa.2018.03.027
0957-4174/© 2018 Elsevier Ltd. All rights reserved.

SDNs are vulnerable and easier to overwhelm (Kreutz et al., 2013; Li, Meng, & Kwok, 2016). Concerns about resiliency and robustness arise when the logic behind forwarding behavior is centralized and located in the controller, which becomes a single point of failure (Sandhya, Sinha, & Haribabu, 2017). Thereby, security threats may include direct controller attacks or vulnerability exploitations in the communication between the controller and the data plane. In the former, the most prominent threat is Distributed Denial of Service (DDoS), which generates malicious requests to overwhelm the controller, consequently hindering the SDN's operation. In the latter, Man in the Middle is a potential attack in which the adversary may break the link between the controller and its switches. To summarize, in the absence of a secure and robust controller, attackers have opportunities to change the behavior of the underlying network by modifying the controller operations.

Thus, it is necessary to adapt the current management model(s) to ensure reliability, resiliency, and network availability. Therefore, it is necessary that network traffic behavior be constantly monitored, aimed at early detection of normal behavior divergences. This allows for rapid decision-making and action against the identified issue. However, conducting an analysis and performing complete monitoring of large-scale network systems are impractical manual tasks for a network administrator. Thus, a traffic monitoring system to detect anomalies should operate in an autonomic manner, aiming to achieve proactive network management.

In this light, the major contribution of this paper is to present an ecosystem able to automatize the task of the administrator in managing SDN networks, performing real-time anomaly detection and mitigation. This particular characteristic ensures reduction of errors and effort arising from human intervention. Our proposal includes routines that perform constant traffic monitoring and control using traffic analysis of packet flows. Such routines are divided into four complementary stages. The first corresponds to the collection and storage of flows through the OpenFlow protocol, which has been widely used to collect network flow statistics passively or actively measure latency by packages injection traffic analysis. The second phase profiles the normal traffic behavior to create a traffic profile or digital signature of the analyzed segments. Current traffic is compared with the previously established traffic profile in order to identify suspicious traffic events. Mitigation of anomaly effects on the network, e.g., neutralization of the detected anomalies composes the third phase. Finally, reports of identified attacks are generated in the last phase to validate and audit system results. We assess the feasibility and the effectiveness of the anomaly detection scheme and mitigation routines in terms of accuracy and system resources usage (CPU usage, flow table size).

The remainder of this paper is organized as follows. Section 2 discusses related work. In Section 3, we present the system design principles. Section 4 describes the results and performance of the system. Finally, Section 5 concludes the paper.

2. Related work

Although Software-defined Networking (SDN) enables new network applications and more flexible control in dynamic network environments, security is still an important concern as it is not yet a built-in feature in the SDN architecture. Increasingly, it is expected that current security schemes will operate in near real time to face a spectrum of threats, inspecting large volumes of traffic, and providing efficient identification of various network anomalies. Thus, we present some previous SDN-related anomaly detection efforts to meet this demand.

Aleroud and Alsmadi (2016) classified the anomalous events that potentially compromise the SDN architecture into three categories: (i) attacks on the control plane, (ii) compromising of communication between the control and data planes, and (iii) threats designed to attack the data plane equipment. A flood attack can directly or indirectly cover these three categories because of the volume of traffic and the increasing number of connection requests. Excessive use of these network resources can overload the controller as well as occupy all the forwarding table entries of the devices in the data plane with flows from illegitimate connections. In this manner, despite the extensive work in SDN security, most of the approaches have been designed to contain flood attacks (e.g., DDoS attacks).

Early work on DDoS detection in the SDN environment was reported by Braga, Mota, and Passito (2010). The presented system continuously observed the statistical features of the flows to identify any anomalous activity. The authors used Self-Organizing Maps (SOM) with the topological neighborhood described by a Gaussian function to classify each packet as benign or abnormal. Kreutz et al. (2013) also focused on exploiting the benefits of SDN to defend the same type of anomaly. However, their target victims reside in the traditional network environment, which makes their solutions unsuitable for SDN. Ha et al. (2016) stated that identifying flood attacks requires detailed processing and inspection of a large volume of traffic. In that light, the authors presented a traffic sampling strategy for SDN networks that maintains the total aggregate volume sampled below the processing capacity of an Intrusion Detection System (IDS). Mousavi and St-Hilaire (2015) proposed an entropy-based mechanism to detect a DDoS attack. In case of an attack, the entropy decreases by evaluating the randomness of incoming packets' destination IP addresses. In Xiao, Qu, Qi, and Li (2015) an effective detection approach based on traffic classification with correlation analysis (CKNN) was proposed. Although the detection rate of known anomalous events is high, such methods become incapable of recognizing variations of the same attack, because signature-based detection is used. For this same reason, the techniques must/should be trained with a large labeled dataset, which is not always possible to obtain. The ecosystem proposed in this paper overcomes these limitations because the detection adapts automatically to changing traffic patterns.

Several studies employ alterations in the SDN infrastructure to conduct anomaly detection. Wang, Zhang, Singh, Lumezanu, and Jiang (2013) proposed NetFuse as a mechanism to protect against traffic overload in OpenFlow-based data center networks. In order to guard the network against the effects of malicious traffic, NetFuse sits between network devices and network controller as an additional layer. Joldzic, Djuric, and Vuletic (2016) presented an alternative solution based on an SDN network topology consisting of three layers to place the IDS. The outermost, located at the network gateway, contains an OpenFlow switch responsible for dividing the incoming traffic and forwarding the generated parts to the intermediate layer. The second layer is composed of several devices called processors, which perform the detection of an attack. In the third layer, the traffic generated by the processors is aggregated and sent to the interior of the network through an OpenFlow switch. Two disadvantages are observed in this approach. First, it assumes that the network is anomaly-free, which can become a problem when anomalies are launched by the internal network itself that make the SDN controller unavailable. The second disadvantage is the division of traffic among the processors, which might lead to the masking of attacks, as due to load balancing, each processor can analyze disconnected parts of the same anomaly. Chen, Junuthula, Siddhrau, Xu, and Chao (2016) applied specialized software boxes to improve the scalability of ingress SDN switches to adjust the control plane's workload during DDoS attacks.

Regarding attack mitigation in SDN, existing solutions limit the rate of requests to the controller excluding those considered surpluses, which also entails the elimination of legitimate requests. To overcome this issue, Wei and Fung (2015) proposed FlowRanger, a controller prioritization buffer to handle routing requests based on their attack probability derived from a trust value of the requestor source. Based on the trust value, FlowRanger sorts the requests into multiple queues of buffers with different priorities. In this manner, a low priority is assigned to potentially malicious requests. Hommes, State, and Engel (2014) introduced a centralized monitoring application that uses variations in the logical topology of the network to detect flood attacks and a basic mitigation rule to react against large traffic bursts. Zaalouk, Khondoker, Marx, and Bayarou (2014) proposed OrchSec, an orchestrator that utilizes network monitoring and SDN control to develop security applications. This architecture can mitigate some attacks; those that do not require a deep inspection of packets contents. In Wang, Zheng, Lou, and Hou (2015), a protection mechanism that relies on a public cloud provider was proposed to detect and mitigate DDoS attacks in SDN. Giotis, Argyropoulos, Androulidakis, Kalogeras, and Maglaris (2014) implemented a mitigation mechanism that drops the packets involved in the anomaly. In the mechanism proposed by Cui et al. (2016), once a DDoS attack is detected, the mitigation module inserts flow entries with the highest priority into the flow table of the switch which is marked as the attack source switch. The intent is to block malicious traffic by dropping packets that match the destination address and the ingress port used in the attack. In Nunes, Schardong, and Schaeffer-Filho (2017), a multi-agent application was proposed to address resilience of SDNs against DDoS attack. Agents may have goals that they are committed to achieving and execute actions (plans) to achieve them. Thus, the plan with the maximum utility to meet the safety of the network is applied. Nevertheless, when a selected plan fails to achieve a goal, the agent delegates the goal to other agents. The authors argue that although the plans are simple, they provide agent coordination, and it is the interplay among decoupled agent parts (rules and plans) that provides the emergent behavior.

In this paper, we focus specifically on the SDN environment, in which the controller is able to collect and analyze traffic statistics reports from switches. Our proposed ecosystem is different from previous work because it does not require any change to the SDN infrastructure and can detect and classify a variety of anomalies in order to choose the correct mitigation strategy. Also, the method provides reports to the network administrator regarding the details of detected anomalies and the efficiency of countermeasures taken against these issues.

3. Overall architecture and design principles

The design of the presented ecosystem for anomaly detection and mitigation in SDN-specific environments is based on the following properties:

- *Autonomic approach:* The proposal enables automation in both network monitoring and detection of anomalous traffic events.
- *Modular design:* Although tasks such as traffic statistics collection, anomaly detection, mitigation of the attacks, and report generation are complementary in our approach, they are designed so that they can be decoupled. It allows any task to be modified without causing interference in another.
- *Efficient use of control plane:* A controller handles the forwarding table of network equipment, installing flow rules in accordance with the classification assigned to each flow.
- *Two monitoring phases:* Unlike methods that analyze the entire aggregate network traffic, our approach monitors the behavior of flows on each switch. The first phase consists of observing and comparing the traffic behavior in relation to what is expected, *i.e.*, its normal profile. If a behavioral deviation is detected, the second phase launches an ostensible monitoring performed at multiple time-windows to identify either the attacker or the targets under attack.

Fig. 1 shows the parts of the ecosystem and how they are related. In an SDN, the data plane consists of various forwarding devices (*e.g.*, switches), which transmit information to its destination by following rules defined by the controller. Request, installation, and update of forwarding rules occur by exchanging messages between the controller and the data plane. In addition, the controller can use some applications implemented by the administrator to manage forwarding rules on the network. Using this feature, we define a set of routines regarding anomaly detection and mitigation. They are grouped into four modules: data collection, anomaly detection, mitigation, and reporting. The figure depicts the iterations among these modules and the activities performed by them.

The controller periodically sends OpenFlow flow-stats request messages to the devices in the data plane to request traffic statistics. Consequently, the devices respond with chunks of content from the flow table. Each chunk contains a portion of the flow-entries (e.g., IP addresses, ports, protocols) along with the packet and bit counters of each flow. After obtaining the response, the controller forwards the messages to the traffic statistics collection module. The next step is to extract the data needed for network monitoring and pre-process this data before passing it to the detection module. In the detection module, it is ascertained whether the newly collected traffic differs from what was expected. If an anomaly is detected, the event type, as well as the switches, IP addresses, and ports involved in the suspected malicious communication are reported to the mitigation module. IP addresses and port information are used to compose the messages sent to the data plane for application of the mitigation routines, where the event type determines the choice of an appropriate mitigation routine. In our approach, mitigation routines include discarding packets from a particular flow or source IP address, or modifying flow-entries to change the routing of packets belonging to a particular flow. Finally, the report module stores information about detected threats, mitigation policies triggered, and addresses and ports involved. Each module is detailed in the following subsections.

Throughout this paper, all vectors are assumed to be column vectors. Lower case boldface letters denote vectors and $\mathbf{x} = [x_1, \ldots, x_n]$ is a column vector. Uppercase boldface letters are used to denote matrices.

3.1. Traffic statistic collection module

This module is responsible for the acquisition of traffic data, which is fundamental to constant network monitoring and is a prerequisite to anomaly detection. Information statistics gathering is accomplished using the OpenFlow protocol. This provides a common interface to control how packets are forwarded by accessing the data plane's internal forwarding tables, configuration, and statistics (Masoudi & Ghaffari, 2016). In addition, it uses the concept of flows to identify network traffic based on pre-defined rules, programmed into the SDN controller software. Flows are defined as a group of packets with one-way transmission sessions that share features such as transport protocol, addresses, and both source and destination ports. Flows are identified by common characteristics, so they are often related to an application, a network device, or a user. As a result, administrators or automated management systems may define how traffic should flow through network devices.

Several studies in the literature explore detecting anomalies using traffic analysis at fixed five-minute intervals (Amaral, de Souza Mendes, Zarpelão, & Proença Jr., 2017). However, due to the continuous increase in transmission rates, monitoring this interval becomes impractical. For example, a network that operates at 10 Gbps may have terabits of information compromised

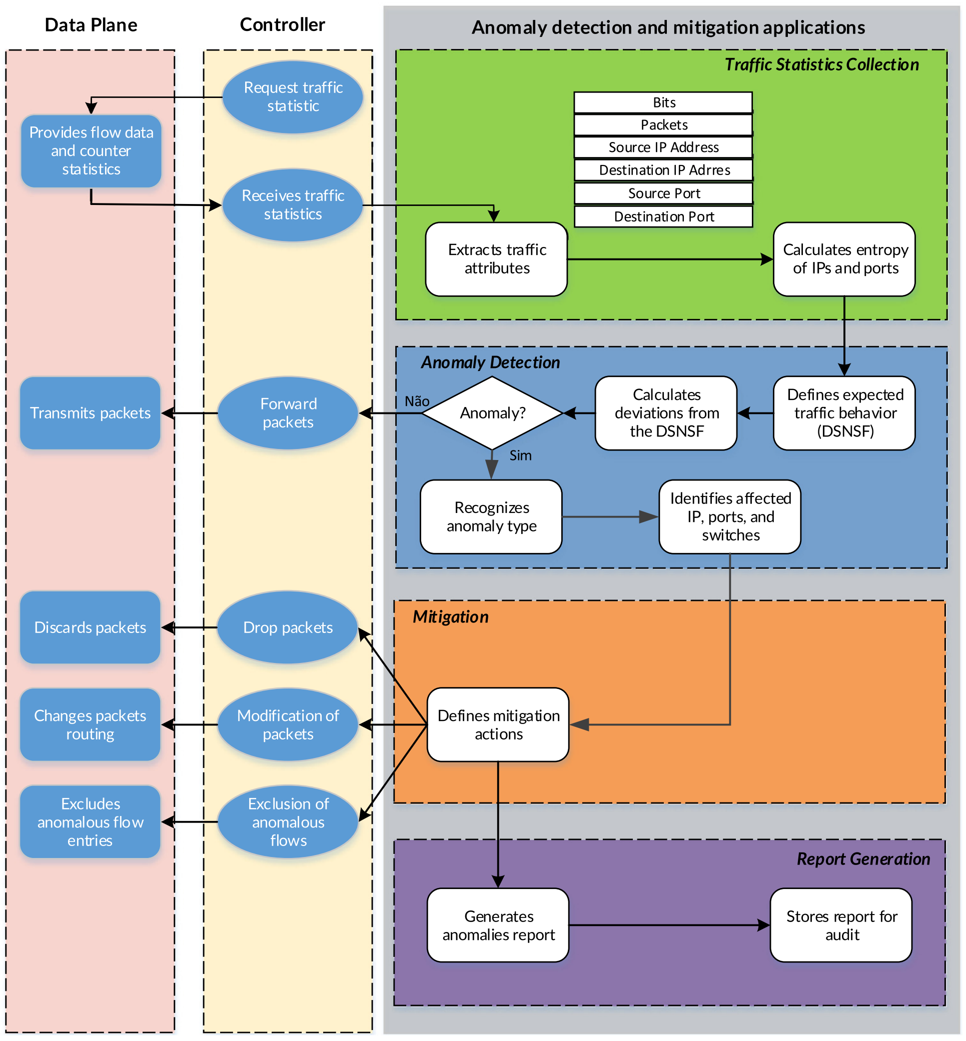

Data Plane
Controller
Anomaly detection and mitigation applications
Traffic Statistics Collection
Request traffic statistic
Provides flow data and counter statistics
Receives traffic statistics
Bits
Packets
Source IP Address
Destination IP Adres
Source Port
Destination Port
Extracts traffic attributes
Calculates entropy of IPs and ports
Anomaly Detection
Transmits packets
Forward packets
Não
Anomaly?
Sim
Calculates deviations from the DSNSF
Defines expected traffic behavior (DSNSF)
Recognizes anomaly type
Identifies affected IP, ports, and switches
Mitigation
Discards packets
Drop packets
Changes packets routing
Modification of packets
Defines mitigation actions
Excludes anomalous flow entries
Exclusion of anomalous flows
Report Generation
Generates anomalies report
Stores report for audit

Fig. 1. Overall ecosystem architecture.

during the five-minute interval. Thus, this proposal deploys a time window of 30 s to request statistics from switches and send them to the anomaly detection module. The choice of this value is to reduce the time response to anomalous traffic events, and triggering countermeasures in near real-time. In this manner, the system ensures fast detection and attack mitigation that otherwise would increase within this period. Once the anomaly subsides, our approach cancels the mitigation to avoid harming legitimate traffic.

The controller periodically sends statistic requests for collection of all flow-entries in the switches, along with their corresponding counters. Upon receiving this request, each switch responds using OpenFlow messages to the requesting controller with chunks of the flow table contents. We define $\Omega = \{\omega_1, \omega_2, \dots, \omega_n\}$ as the set of flow information sent by a switch to the controller, and each w_i contains the following properties: (i) Received packets that measure the amount of packets matching the flow they belong to; (ii) Received bits representing the total bits carried by a flow; (iii) Source IP address; (iv) Destination IP address; (v) Source port; (vi) Destination port and (vii) transport layer protocol. The system processes the data collected on each switch apart from the information acquired from other switches. Thus, the method for identifying the OpenFlow switch affected by the anomaly is straightforward.

In our previous works (de Assis, Hamamoto, Abrão, & Proença, 2017; Fernandes Jr., Carvalho, Rodrigues, & Proença Jr., 2016), we have conducted extensive statistical analysis to profile the normal patterns of high-speed traffic links. As a result, we found that the observed nature of networks traffic is dynamic and diverse, and also that the amount of information transmitted and its composition changes with time and/or day. From this diverse traffic mix, some traffic attributes were selected to study the behavioral aspects. The selected attributes are extracted every time interval from the Ω set, and they consist of six traffic features, $F = \{f_1, f_2, \dots, f_6\}$, as shown in Table 1.

Table 1
Extracted traffic features from the Ω set.

i	Feature (f_i)	Description
1	Bits	Number of bits transmitted
2	Packets	Number of packets transmitted
3	H(srcIP)	Source IP entropy
4	H(dstIP)	Destination IP entropy
5	H(srcPort)	Source Port entropy
6	H(dstPort)	Destination Port entropy

The bottom four traffic properties are qualitative metrics extracted from the packet header of each ω_i. These are relatively stable with time because at any moment they hold intrinsic correlations among themselves. The benefits of these qualitative attributes are twofold. First, they provide information to recognize devices and applications operating suspiciously. Second, they become sensitive indicators of traffic behavior changes when a statistical model is used to calculate their distribution.

For the purposes of this paper, each feature must be represented by a quantitative value. For bits and packets (volume attributes) it is assumed their quantities are collected in each analyzed time window, and for IP addresses and port features, we enable their use in traffic analysis by adopting the Shannon Entropy concept. Given a set of measurements of the attribute $A = \{a_1, \ldots, a_n\}$, in which a_i represents the occurrences number of sample i, the entropy H for the attribute A is defined as:

$$H(A) = -\sum_{i=1}^{n} p_i \log_2 p_i, \qquad (1)$$

where $p_i = a_i/n$ is the occurrence probability of a_i. On the one hand, when sample distributions are concentrated, the entropy value is minimal, and is zero when all samples are identical. On the other hand, when the dispersion increases for a given traffic feature A, then $H(A) \le \log_2 n$ holds with equality *iff* $p_i = 1/n$ for all i.

Entropy calculation summarizes the information related to IP addresses and ports over a time window, representing the dispersion or concentration degree of each attribute with a quantitative value. Thus, under traffic anomalies, there will be detectable changes on the distributional aspects of these chosen traffic features. For example, during a DDoS attack, the source IP addresses distribution may become dispersed due to many hosts triggering a flood attack. The same occurs to destination ports when they are scanned to find vulnerabilities. In contrast, when a particular source establishes a high number of connections, the source IP distribution becomes concentrated.

3.2. Anomaly detection module

Our anomaly detection problem can be stated as follows. A stream of data points $\mathbf{x}$ in $\mathbb{R}^k$ consists of a set of measurements $\{x_t\}_{t=1}^{T}$ regulated by a probability distribution P. Although all measures are consequences of specific events within the space of events S, the mapping $f : S \to \mathbb{R}^k$ may not be previously known. Initially, one assumes that S can be divided into two or more subspaces corresponding to normal and the anomalous network traffic conditions. Also, it is necessary to infer the membership of a particular event in one of the subspaces using the corresponding measurement. The former is accomplished by the traffic characterization task and the latter by anomaly identification as described in the following subsections.

3.2.1. Traffic characterization

Identification of unusual traffic patterns is an important activity in network management and anomaly detection. Among different methodologies, the creation of the network behavior profile stands out for providing dynamic traffic monitoring and detection. Accordingly, more attention has been focused on these methods as they can identify even unknown types of anomalies (Bhuyan, Bhattacharyya, & Kalita, 2014; Pena, Carvalho, Barbon Jr., Rodrigues, & Proença Jr., 2017). With traffic profile knowledge, it is possible to continuously monitor the network to spot time windows in which current traffic appears to be anomalous.

The traffic characterization module performs a characterization of each OpenFlow switch to extract common events represented in the historical traffic measurements. Such process is accomplished for all collected features presented in Table 1, and its outcome is a normal traffic profile represented by matrix $\mathbf{D}$ with dimensions $n \times |F|$, where n is the total time window of a day and $|F|$ is the data dimensionality, or the number of traffic features. Therefore, d_{ij} indicates the expected value of the jth feature at time interval i. In this paper, $\mathbf{D}$ is also referred as the Digital Signature of Network Segment using Flow analysis (DSNSF) (Proença, Coppelmans, Bottoli, Alberti, & Mendes, 2004). Subsequently, the normal traffic profile is compared to the current traffic so that actions are triggered to prevent propagation of anomalies according to their type (*e.g.*, one of the subspaces of S). It is important to note that network monitoring and traffic characterization are concurrent, because measurements of recent past time windows are automatically embedded in the historical traffic for later processing.

After ensuring that all collected traffic features are represented by quantitative values, the extraction of normal behavior begins. The traffic profile creation is built upon our previous clustering technique that provides a basis for anomaly detection. This technique is named Ant Colony Optimization for Digital Signature (ACODS) (Carvalho, Barbon Jr., de Souza Mendes, & Proença Jr., 2016). It is a modification of traditional ACO metaheuristic, and clusters the traffic to discover sensible organization of each historical traffic time window. According to Jiang and Papavassiliou (2006), the ants' habit of living in groups is essentially similar to the grouping of data. Algorithms based on ant behavior have natural advantages in the application of cluster analysis, such as self-organization, flexibility, robustness, absence of need for prior information, and decentralization. In addition to the previous benefits, we advocate the use of ACODS because it is a supervised learning technique, which allows the characterization of traffic to occur in an autonomic way. Also, it leverages the construction of solutions not given by local optima, which is an existing problem in some clustering algorithms.

In ACODS, the simulation of foraging behavior of ants combined with the ability to find the shortest path between their colony and food is used to find the cluster set. Initially, the analyzed attributes are given by a six-tuple composed of bits and packet counters followed by the entropy calculated for the source IP, destination IP, and source and destination port. Ants, using statistics and probabilities, travel through the search space represented by a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, in which $\mathcal{V}$ is a finite set of all nodes and $\mathcal{E}$ is the edges set. We assume that the paths are formed between the center of a cluster (centroid) and each six-tuple that will be clustered. Ants are attracted to paths that lead to the most promising cluster solutions. In ACODS, a better solution minimizes the intra-cluster distance, i.e., the sum of the distance between objects and the centroid of the cluster they belong to, given by:

$$J = \sum_{j=1}^{k} \sum_{\mathbf{x}_i \in C_j} dist\left(\mathbf{x}_i, \bar{\mathbf{x}}^{(j)}\right)^2, \qquad (2)$$

where k is the number of clusters, $\mathbf{x}_i$ is the ith six-tuple, $\bar{\mathbf{x}}^{(j)}$ is the center of cluster C_j, $dist\left(\mathbf{x}_i, and\ \bar{\mathbf{x}}^{(j)}\right)$ is the Euclidean distance between $\mathbf{x}_i$ and $\bar{\mathbf{x}}^{(j)}$.

The activities carried out at each iteration for DSNSF creation are classified into three categories:

Build solution: Ants move from one node to another by assigning each object to a certain cluster.

Local search: It tests and validates a solution created by ants. A local search may be applied to assist in building good solutions. In our design, local searches are used to change an unpromising portion of solutions (paths that lead to poor solutions).

Pheromone update: The ants deposit pheromone as they walk through the graph and, as a consequence, paths frequently used for the solution construction contain a higher concentration of pheromone. The increased concentration of pheromone is an essential factor in the algorithm implementation because it directs ants to seek new paths more prone to acquire a near-optimal solution. On the other hand, less promising paths are slowly forgotten through evaporation of the pheromone. This pheromone decreasing

Table 2
Anomalies and their effects on traffic features. Adapted from Carvalho et al. (2016).

k	Anomaly	Affected traffic features					
		Bits	Packets	H(srcIP)	H(dstIP)	H(srcPort)	H(dstPort)
1	DDoS	n.c.	+	n.c.	−	+	−
2	Portscan	n.c.	+	−	−	+	+
3	Flash Crowd	+	+	+	−	−	−

technique facilitates acceptable convergence time and contributes to the construction of non-premature solutions.

After clustering, the traffic collected in an interval is separated into groups, where each six-tuple is more similar to the data of its group than any other. Owing to the high similarity of network traffic, most information represents similar behavior and some dense regions in the research space are formed. As previous works have pointed out (Ahmed, Mahmood, & Hu, 2016), in a clustering with clusters of various sizes, the smaller and sparser clusters can be considered anomalous and the thicker ones are considered normal. Instances belonging to clusters with sizes below a threshold are considered unusual, and they are discarded from normal traffic patterns.

Assume the output of the traffic profiling, matrix $\mathbf{D}$ represents the expected traffic behavior for an entire day, and vector $\mathbf{d}_t = [d_1, d_2, \ldots, d_{|F|}]$ defines the expected normal behavior of the six analyzed features at a given time instant t. The traffic sampled at t is given by F_t and it is acquired from periodic statistic requests sent to the switches, as discussed in Section 3.1. In this manner, a suspicious event is detected by deviation of ongoing monitored traffic F_t in relation to the normal expected profile $\mathbf{d}_t$.

3.2.2. Anomaly identification

Although alarms are generated when normality deviation occurs in any one of the six analyzed attributes, a general alarm is triggered when observing any anomalous traffic as summarized in Table 2. The table lists anomalies commonly found in backbone traffic and their effects on flow measurements. An index k is assigned to each anomaly which our system can recognize. The symbol "+" indicates that traffic feature measurement increases during the anomaly, "−" represents decreasing traffic, and features with notation "n.c." do not change with the threat occurrence.

In Table 2, each anomalous event has its own effect on traffic attributes. The relevance of each of these attributes for the correct identification of an attack should be assigned. Regression models are suitable candidates for weighing the traffic changes that lead to the detection of an event. In a simple or multiple linear regression model, the dependent variable Y is a continuous random variable. However, in some situations, the dependent variable is qualitative and is expressed by two or more categories, admitting two or more values. Because of this, the least squares method does not provide feasible estimators, while logistic regression allows the use of a regression model to calculate or predict the probability of an event.

The multinomial logistic regression (MLR) model is a specific case of logistic regression that is used for data in which the dependent variable is unordered or polytomous, and independent variables are continuous or categorical predictors (Wang, 2005). MLR designs a classifier capable of distinguishing K classes using labeled training samples. In this study, MLR computes the importance of variations in the traffic measurements to diagnose anomalies, and based on this variation assigns a likelihood ratio for each state in the space of events S.

Given a set of $r+1$ independent variables denoted by $\mathbf{X} = \{X_0, X_1, \ldots, X_r\}$, assuming values $\mathbf{x} = [x_0, x_1, \ldots, x_r]$ with $x_0 = 1$, and a random variable Y that can assume values $y \in \{0, 1, \ldots, q\}$. Unlike a binary logistic model, in which a dependent variable has only two choices (*e.g.*, normal or anomalous), the dependent variable in a multinomial logistic regression can have more than two choices arranged categorically, where one of these categories represents the reference category. In this paper, $k = 0$ ("normal") is the reference category, and $k = 1$ ("DDoS"), $k = 2$ ("portscan") and $k = 3$ ("flash crowd") are the remaining categories. Given the aforementioned categories, one can describe the logit function (or dependence) comparing $Y = k$ and $Y = 0$ for $k \in \{1, \ldots, q\}$. The logit function is denoted by:

$$\begin{aligned} g_k(\mathbf{x}) &= \ln\left[\frac{P(Y_i = k|\mathbf{x})}{P(Y_i = 0|\mathbf{x})}\right] \\ &= b_{k0}x_{k0} + b_{k1}x_1 + \cdots + b_{kr}x_r \\ &= \mathbf{x}^\top \mathbf{b}_k, \qquad k \in \{0, \ldots, q\}, \end{aligned} \tag{3}$$

where $\mathbf{b}_k = [b_{k0}, \ldots, b_{kr}]^\top$ is a vector of unknown parameters, *i.e.*, weights assigned to traffic features and $x_{k0} = 1$.

From Eq. (3), one can apply logarithm properties to obtain:

$$\exp\left[g_k(\mathbf{x})\right] = \frac{P(Y_i = k|\mathbf{x})}{P(Y_i = 0|\mathbf{x})}, \qquad k \in \{0, \ldots, q\}$$

$$P(Y_i = k|\mathbf{x}) = \exp\left[g_k(\mathbf{x})\right] P(Y_i = 0|\mathbf{x}). \tag{4}$$

Using the fact that all $q+1$ probabilities must sum to one, we find:

$$P(Y_i = 0|\mathbf{x}) = 1 - \sum_{k=1}^{q} P(Y_i = 0|\mathbf{x}) \exp[g_k(\mathbf{x})] = \frac{1}{1 + \sum_{k=1}^{q} \exp[g_k(\mathbf{x})]}. \tag{5}$$

Substituting Eq. (5) in Eq. (4), we can determine the other probabilities:

$$\pi_k = \frac{\exp[g_k(\mathbf{x})]}{1 + \sum_{k=1}^{q} \exp[g_k(\mathbf{x})]}, \tag{6}$$

where π_k denotes $P(Y_i = k|\mathbf{x})$.

After defining the normal behavior and how unusual events are identified, we state the anomaly classification problem in mathematical fashion as follows: Given a time bin t, an ongoing traffic measurement F_t is assigned to the class k if

$$k = \arg\max_{\ell} (\pi_\ell(\boldsymbol{\Theta}_t)), \tag{7}$$

were $\pi_\ell(.)$ is a likelihood ratio assigned to each anomaly ℓ, which is properly identified by the detection mechanism. The vector $\boldsymbol{\Theta}_t$ is given by:

$$\Theta_t = \boldsymbol{\Lambda}^{-1}|\mathbf{d}_t - F_t| \tag{8}$$

in which $\mathbf{d}_t$ is the expected profile, F_t corresponds to the current traffic measurement, and Λ^{-1} is the inverse of a diagonal matrix whose elements are F_t. Each value $\boldsymbol{\Theta}_t = \left[\theta_{t1}, \theta_{t2}, \ldots, \theta_{t|F|}\right]$ denotes the relative error between the expected and current traffic measurement for each traffic feature.

Although the approach can detect the anomalies shown in Table 2, there are some legitimate network abnormalities which affect the traffic features in the same manner as a malicious anomaly. To avoid interruption of benign traffic events, the detection system maintains a whitelist to store IP addresses and ports whose unusual behavior may actually be anomaly-free traffic. Thus, the module first checks for anomaly-related IP addresses and ports in the whitelist and then mitigation routines are applied if necessary. The whitelist can be maintained by an automated procedure or set manually by the network manager. We implemented the latter for simplicity.

When an anomaly is recognized, the system conducts more active analysis over suspected flows and in suspected network regions, where IP addresses and ports that compose an unusual scenario are inspected during multiple time windows. Addresses identify hosts involved in the anomalous communication while ports indicate services which may be compromised. The total packets and bits sent and received by each of these hosts are monitored as well as the number of connections established by them. If these values are contributing to the deviation from the expected behavior of the switch, the mitigation module takes countermeasures to return the network to its normal operation.

All the procedures required for anomaly identification are summarized in Algorithm 1. Each step is arranged so that the MLR classifier can identify anomalies in SDN network traffic after weight training. Once the anomaly detection module receives the data from the traffic statistics collection module, the relative error between the collected traffic F_t and the expected behavior $\mathbf{d}_t$ is calculated and stored in $\mathbf{\Theta}_t$. Then, the logit function is calculated using the previously calculated error and the weights obtained from the MLR training. In line 3 of the pseudocode, we obtain the probability of the collected traffic belonging to the normal class, and the probability of the traffic being considered a DDoS, portscan, or flash crowd class anomaly is computed. In line 5, the algorithm assigns to the analyzed traffic the class with the highest degree of pertinence and, finally, the label of the class is returned in line 6. The outcome of the pseudocode for each time interval is shown in Fig. 4(a), and according to these outputs, the effectiveness of the detection method can be assessed as shown in Figs. 5 and 6.

Algorithm 1 Anomaly identification.

```
Input: current traffic measurements $F_t$, expected traffic measurements $\mathbf{d}_t$
Output: index of the anomaly $k$
1: Compute $\mathbf{\Theta}_t$ using $F_t$ and $\mathbf{d}_t$                         ▷ Eq. (8)
2: Compute the logit function using $\mathbf{\Theta}_t$                          ▷ Eq. (3)
3: Calculate the probability of $\mathbf{\Theta}_t$ being the reference class     ▷ Eq. (5)
4: Measure the probability of $\mathbf{\Theta}_t$ being in anomaly class $k \in \{1, \dots, q\}$ Eq. (6)
5: Assign $\mathbf{\Theta}_t$ to the anomaly class $k$ with the highest calculated probability   ▷ Eq. (7)
6: return $k$
```

3.3. Mitigation module

Anomalies can harm network operation, so, countermeasures should be taken to nullify any event that may affect the quality of services provided to customers. The central points of a resiliency strategy are the management and the reconfiguration of detection and restoration mechanisms, which function as autonomous components in the network infrastructure (Nunes et al., 2017). While anomaly detection efforts such as monitoring and traffic classification provide the recognition and categorization of attacks, repair mechanisms can be used in the mitigation of these threats. In this manner, the proposed system implements routines for locating flows and equipment causing or being affected by anomalies and new policies are applied to them.

Table 3
List of implemented policies.

Policy	Description
Block_src_IP	Blocks the traffic of a misbehaving IP address
Block_flow	Drops packets that belong to a particular flow
Load_balance	Traffic is distributed to other servers during intensive network activity

Policies are a collection of rules that determine how present and future decisions are taken. Policy actions are performed when specific events occur and certain conditions are met. In this manner, policies are defined as an *Event-Condition-Action* (ECA) model, which is suitable for dynamic policy management (Sahay, Blanc, Zhang, Toumi, & Debar, 2017). Each *Event* refers to a specific attack or incident and is associated with a set of rules. The rules are described as a set of *Conditions* that match the context in which the attack or incident occurs. Finally, the *Action*, is essentially a high-level action to be applied to the identified flows.

The use of policies can provide three benefits (Han & Lei, 2012). First, administrators predefine policies and store them in a repository. When an event occurs, the system requests and accesses these policies automatically, without manual intervention. Second, the formal description of policies allows autonomous analysis and verification of consistency to some extent. Third, by using abstract technical details, policies can be inspected and changed at runtime without modifying the system implementation.

A list of policies implemented is shown in Table 3. In short, our focus is on enforcing the policy to mitigate anomalies once they are detected. For this purpose, flow entries containing the IP address and port learned in the anomaly detection module are inserted into the forwarding tables of switches along with actions that must be performed on the new packets that match these flows. The main actions include forward, drop, or modify packet fields (ONF, 2009). The *block_src_IP* policy cuts off incoming traffic from a malicious host (*e.g.*, portscan attack) while *block_flow* blocks the communication from a certain host to a specific service in the destination IP address. The mitigation module attaches a forwarding action in *load_balance* to distribute network traffic loading generated by a DDoS attack through multiple servers to avoid degrading services performance. In order to promote failover of high-availability servers, the network administrator can indicate which other servers can perform the same task as an overwhelmed server. This policy uses a Round-Robin algorithm to ensure that all servers used to balance the load will be visited in a loop, determining which server to use according to the last selected one. After the countermeasure is designed, the mitigation module pushes the policy to the switch through the network controller using the OpenFlow protocol, which acts directly in forwarding logic to the network equipment.

Flow entries corresponding to mitigation rules are assigned with higher execution priority than other active entries in the forwarding table. Furthermore, it is assigned an amount of time (idle timeout) so that these flows remain active when a match no longer occurs, allowing the mitigation to continue as long as the anomaly is active.

3.4. Reporting module

According to Bhuyan et al. (2014), a weakness of profile-based detection is that anomalies can be slowly introduced in usual patterns until they become a legitimate event. The presented system offers the administrator relevant information to help determine a solution to such issues. This information is arranged in reports, and demonstrates the use of network resources at the moment an anomaly is detected. If these anomalous events are identified

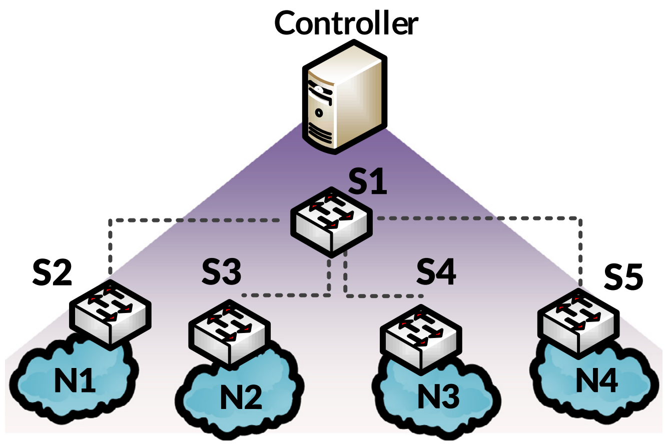


Fig. 2. Tree-like topology emulated on Mininet.

and interrupted, they will not be included in the historical dataset, and consequently, they will not be included in the normal traffic profile (DSNSF). Furthermore, these reports will assist the network administrator to develop new policies to reduce the impact on the services offered to end users.

4. Results

This section evaluates the effectiveness of the proposed system. First, we demonstrate the ability to detect anomalies and their correct classification. The analysis also compares the detection strategy with other well-known classification techniques. Second, we analyze the results of the mitigation module designed to disrupt the attacks that undermine the proper functioning of the network.

4.1. Evaluation settings

To evaluate the performance of the proposed system, we conducted network emulation scenarios using Mininet (Lantz, Heller, & McKeown, 2010), which creates a network of virtual hosts, switches, controllers, and links. Applications developed on Mininet can be moved to a real environment with minimal changes for real-world deployment. The emulated environment created is a tree-like network with depth two, as depicted in Fig. 2. All switches are interconnected by the root switch and each subnet (N1, N2, N3, and N4) is composed of twenty hosts. In order to conduct the experiments, Open vSwitch was used for network switches, as it can handle the required traffic and supports the OpenFlow protocol. The system's modules (statistic collection, anomaly detection, and mitigation) implement all the necessary algorithms in the POX controller, a Python-based platform for the development and prototyping of SDN applications.

The methodology assumed in this paper creates a normal network profile based on the historical traffic. In this manner, we employed Scapy (2017), a programmable tool that sends packet sequences with arbitrary protocol field values (e.g., IP address, ports, and content) to generate historical traffic. The test scenario was built to mimic real environments, being as realistic and reliable as possible by taking into account suggestions proposed by Cui et al. (2016). The normal traffic transferred in the test was composed of 85% TCP traffic, 10% UDP, and 5% ICMP. Meanwhile, the packet size and the rate of generated traffic obey a uniform distribution.

Aiming to test the system effectiveness, two synthetic anomalies were generated to mimic certain attack behaviors. The overall analysis included mixing diverse types of legitimate traffic and varying attack traffic parameters. The malicious traffic was produced by hping3 (2006), a TCP/IP packet assembler and analyzer mainly used as a security tool. To emulate a DDoS attack, hping3 sends synchronize (SYN) packets to a host from a random and constantly changed set of source IP addresses and ports. The second anomaly corresponds to a portscan attack, where a source sends packets to different ports of the destination host, aiming to receive confirmation whether they are operative. Both attacks were directed to a host in N4 whose IP was 10.0.0.78.

In order to train the multinomial logistic regression classifier, samples of attacks conducted in a real network were used (Hamamoto, Carvalho, Sampaio, Abrão, & Proença, 2018). Three types of attacks have been created of varying intensity: DDoS, portscan, and flash crowd. Each observation in the training set passed through all the stages of multinomial logistic regression,

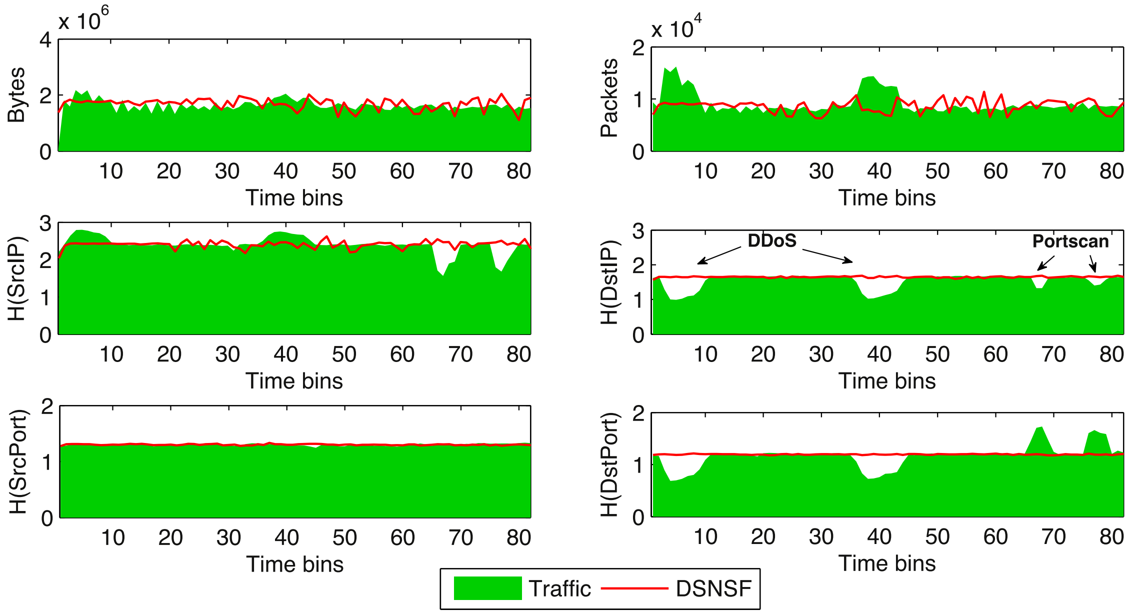
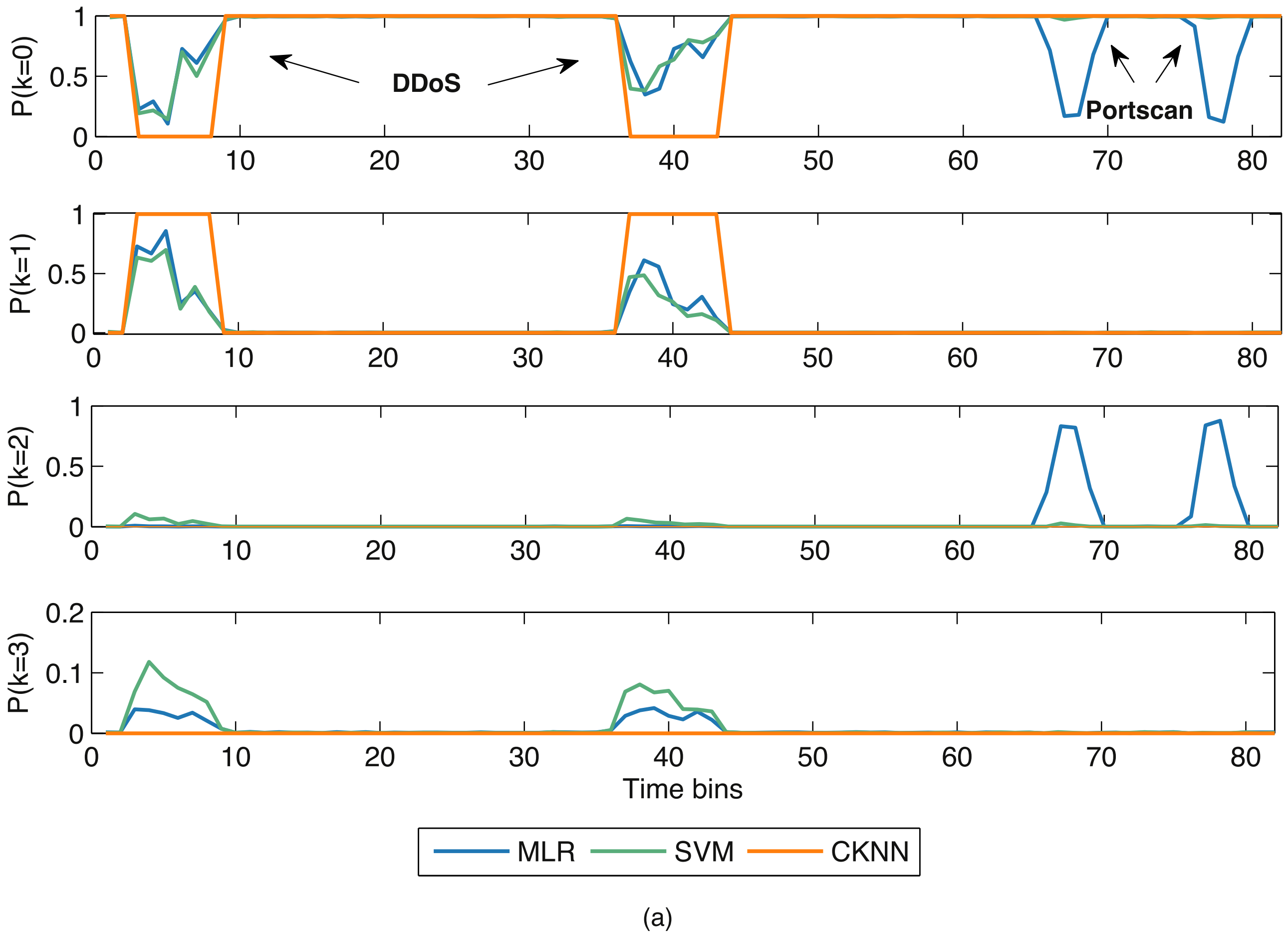


Fig. 3. Traffic characterization using ACODS. The graphs show the normal traffic prediction calculated for each analyzed attribute.



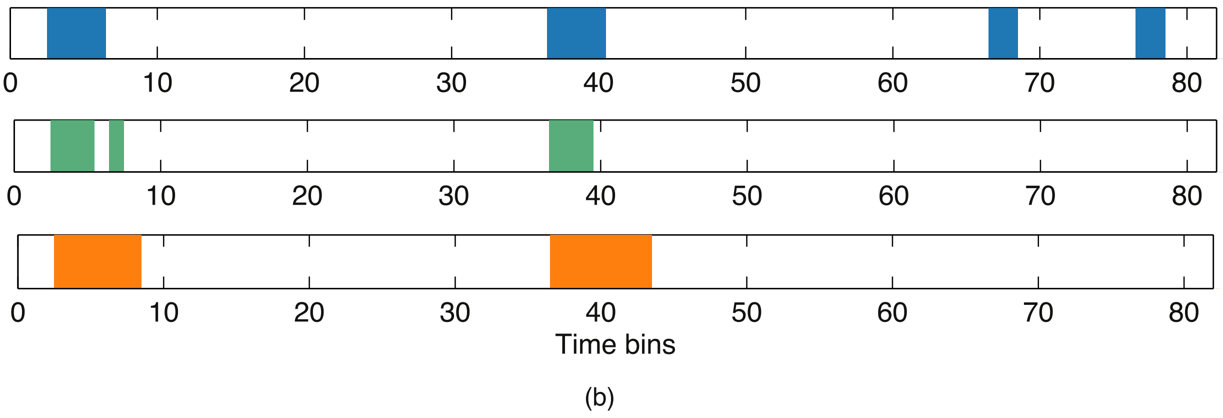


Fig. 4. (a) Probability of each time interval being classified as class k. (b) Alarms triggered by each anomaly detection scheme.

and the proper weights $\mathbf{b}_k = [b_1, b_2, \ldots, b_3]$ were computed using a Limited-memory BroydenFletcherGoldfarbShanno (BFGS) optimization algorithm. The feasibility of $\mathbf{b_k}$, from Eq. (3) is calculated for each observation $\mathbf{x}$ in the training data using a cost function. This function measures the distance between the predicted class using the calculated weights and the actual target class. Adjusting $\mathbf{b}_k$ is an iterative process and the optimization procedure ends when the cost function value is negligible. The output after training the multinomial logistic regression classifier are the calculated weights that allowed correct classification of the samples of the training dataset.

4.2. Anomaly detection evaluation

Fig. 3 shows the data collected from switch S5 during the anomaly detection strategy evaluation. The green area depicts the observed traffic and the DSNSF generated for each attribute is represented by the red line. Two DDoS and portscan traces were generated. The DDoS attacks are differentiated by their intensity. The first attack, presented in the first time bins, caused an unusual increase in packet arrival of 70%, while the second (around time bin 40) increased arrivals by approximately 55%. To achieve these values, 300 and 250 attackers were used in the first and second

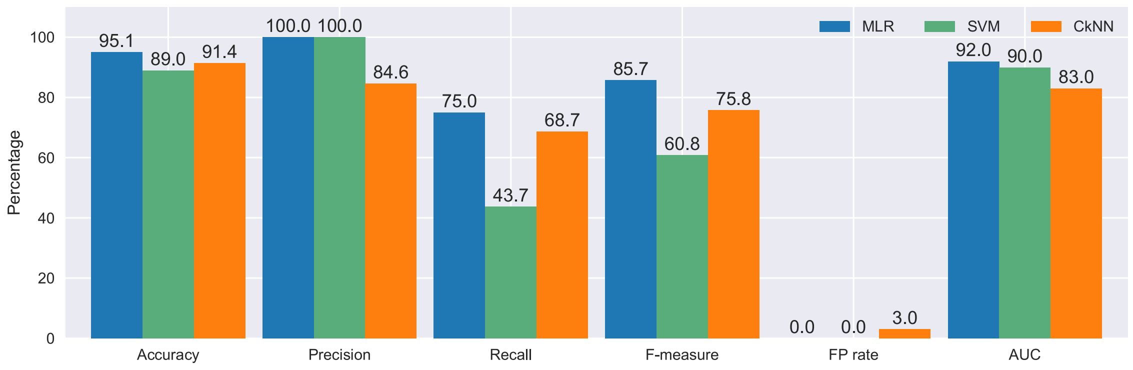


Fig. 5. Anomaly detection comparison among MLR and two popular methods.

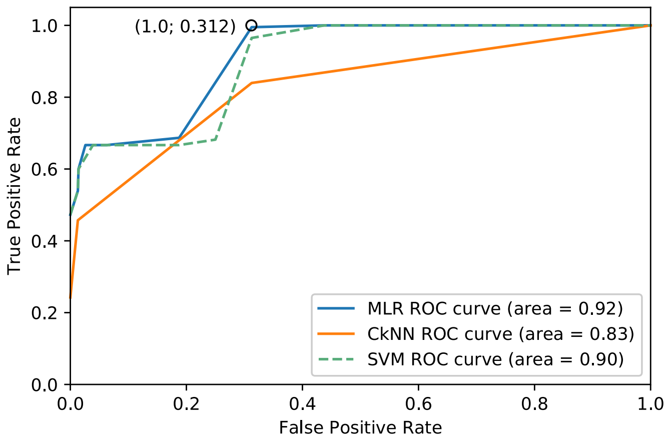


Fig. 6. ROC curves of the compared approaches for anomaly detection.

DDoS events, respectively. Regarding portscan anomalies (around time bins 70 and 80), only one attacker was used to access ports 1–1024 of the victim host. The difference between the attacks was the packet arrival interval, which was set as 0.03 and 0.05 for the first and second portscan attacks, respectively.

4.3. Comparison with other anomaly detection approaches

To further validate our system, we compare our proposal with other anomaly detection methods. The first is k-nearest neighbor (kNN) traffic classification with correlation analysis (CkNN) (Xiao et al., 2015), proposed to detect attacks with high efficiency and low cost in data center networks. CkNN exploits correlation information of training data to improve the anomaly flow classification and reduce the overhead caused by the density of training data. In particular, the k-nearest neighbor algorithm is used to group flows according to recognized malicious events. The second method is based on the support vector machine (SVM) classification technique (Phan, Toan, Tuyen, Huong, & Thanh, 2016) designed for flooding attacks detection. To do so, a kernel function implicitly maps the input space to a higher dimensional feature space to create a clearer separation between normal and anomalous data.

Fig. 4(a) presents the probability at each time interval that the compared classifiers assign to each class. The classifiers assign the class with the highest confidence. Note that our proposed multinomial logistic regression (MLR) classifier was the only approach that presented a high probability of correct classification in the intervals where the portscan attacks were carried out. In contrast, even when DDoS attacks reached values above 50% using MLR or SVM, the probabilities of classifying the correct class were lower than obtained by CkNN. Although the flash crowd anomaly was present only in the training set, it is interesting to analyze the probability of classification of this event. SVM assigned up to 10% and 6% probability of DDoS attacks being flash crowd attacks. Similarly, MLR obtained lower probability while CkNN pointed out no chance of Flash Crowd occurrence.

Fig. 4(b) depicts the time bins in which each classification approach detected anomalies. MLR recognized all the attack traces while SVM and CkNN identified only the DDoS attacks. Once a robust classification model is developed, it is necessary to decide whether it is sufficient to detect the investigated anomalies. A detailed analysis is presented in Fig. 5, which presents the performance metrics comparing the classification approaches. As can be seen, MLR achieved a better overall performance compared to the other methods. CkNN performed worse than SVM in precision and False-Positive (FP) rate. By means of an in-depth analysis, it was possible to verify that these measures were related to the misinterpretation of the CkNN for traffic at the end of the DDoS attacks. While the other two methods classified only the time intervals when such anomalies were in effect, CkNN generated alarms after traffic had already normalized. As expected, Recall was the measure with the lowest values because both CkNN and SVM methods were not able to recognize the time bins where portscan anomalies occurred. It is worth noticing that even though unable to identify the attacks during their entire duration, MLR and SVM did not misclassify a non-anomalous interval.

Fig. 6 shows the receiver operating characteristic (ROC) curves for the compared detection techniques. ROC curves are typically used in binary classification (*i.e.*, normal or anomalous) to assess the output of a classifier. Macro-averaging for true-positive and false-positive measures were used to extend ROC curves to our multi-class scenario. This approach gives equal weight to the classification of each class in a one-versus-all classifier. As expected, MLR performed better because it achieved a larger rate of true-positives with fewer false-positives than the other techniques. MLR could achieve 100% accuracy with a false-positive rate of 31.2%.

4.4. Performance and mitigation evaluation

Once the anomaly detection algorithm has detected and identified a suspicious activity as a DDoS or a portscan attack, the mitigation module can properly handle the corresponding anomalous traffic. Fig. 7 depicts the traffic network behavior when mitigation policies are applied in contrast with its absence. The area comprises the traffic generated following the methodology described in Section 4.2 along with the DDoS and portscan attacks previously used in Fig. 3. Note that when the attack is mitigated, the feature values fall into their expected ranges, *i.e.*, normal behavior. The blue line corresponds to the *block_flow* policy to mitigate the DDoS attack. We also performed tests using the *load_balance* policy, but it presented no significant changes in feature values be-

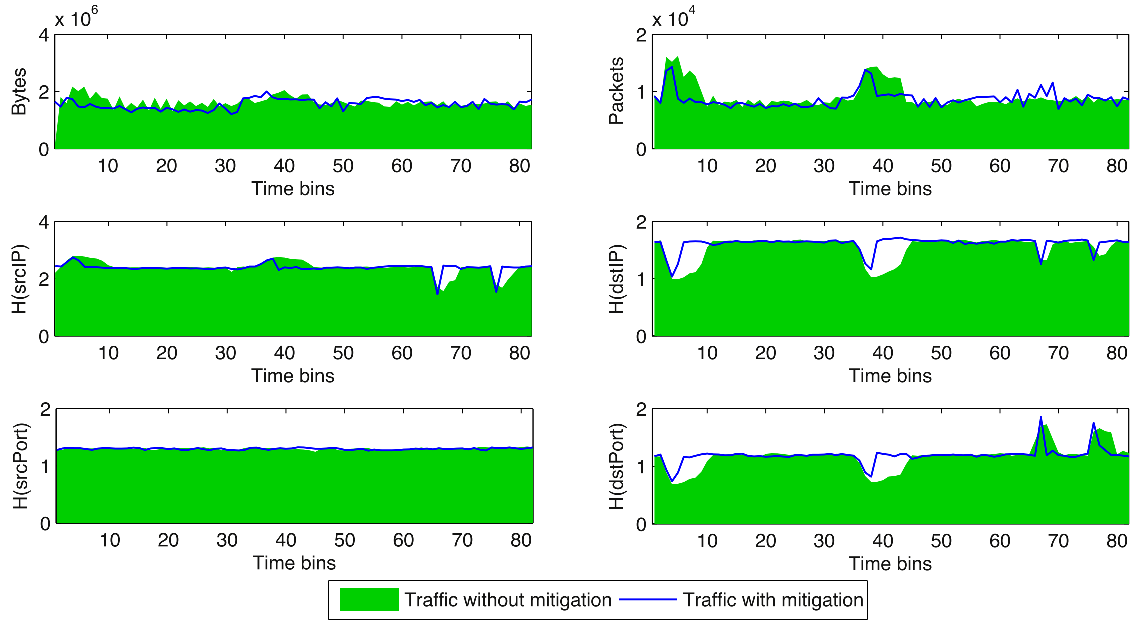


Fig. 7. Traffic feature values during DDoS and Portscan attacks with and without the mitigation mechanism.

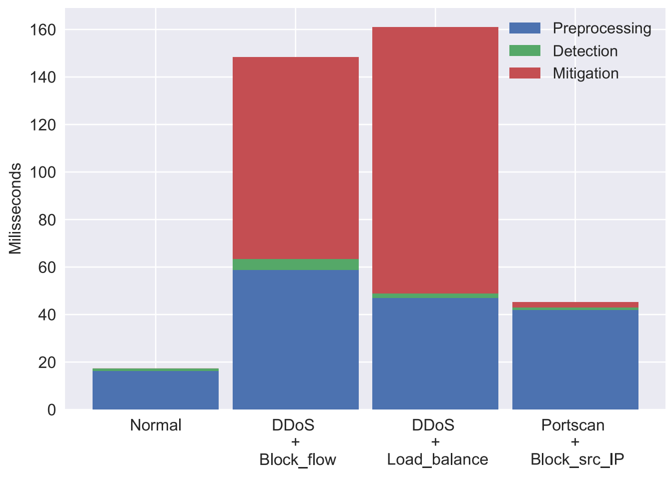


Fig. 8. Time spent on each task performed by the system.

cause it does not alter the rate of requests arriving at an attacked host. Therefore, the result obtained by such policy is similar to traffic without mitigation. However, with regard to network resilience, the *load_balance* effects can be decisive by adoption of this policy. Regarding portscan attacks, the system applied the *block_src_IP* policy. As can be observed, its mitigation required fewer time bins because recognition of the malicious source IP affecting the attacked host is simpler than identifying misbehaving hosts during a distributed attack such as DDoS.

After evaluating the detection and mitigation capacities, the proposed system's overall performance is addressed. Fig. 8 shows the amount of time spent to execute the system's routines according to the detected anomaly and mitigation policies taken. In the figure, the preprocessing procedure aggregates statistic collection and entropy calculation tasks. Mitigation corresponds to the time spent to assign source(s) and destination(s) of a detected anomaly and to push policies to affected switches through the network controller. It is important to note that the communication time between the switches and the controller is not taken into account.

The anomaly module exhibited similar execution time in all analyzed situations. Once the MLR is trained, the traffic assessment and classification is straightforward. The more significant variation of the time period spent by this module is related to the discovery of the IP addresses and ports involved in the anomalous event. Regarding the mitigation module, a determinant time factor is the number of flows that must be inserted in the switches flow table for policy compliance, either by blocking the communication or applying load balancing. The latter takes longer to perform due to the need for determining which server requests will be forwarded and to which alternate servers traffic will be diverted to. The preprocessing phase consumed much of the system's processing time in all analyzed situations, and this value is highly dependent on the intensity of an attack.

We investigate the results of the mitigation module in relation to controller central processing unit (CPU) usage and number of flow-entries in the switches. Both values can be used to measure the system operation. CPU usage indicates the performance of the control plane, while the number of flow entries in the table of a switch can affect the forwarding capability because every incoming packet is matched against flow table entries (Yuan et al., 2017). The three graphics at the top of Fig. 9 represent the controller's CPU usage. These measurements were taken every second for a period of five minutes. Each graph refers to a policy applied to contain an attack, and the red circles indicate the exact moment when such policies were pushed from the controller to the switches affected by the attacks. The three graphs at the bottom of the figure indicate the number of flow entries contained in switch S5 during the attacks.

Fig. 9(a) reveals that the *block_flow* policy required three time bins to restrain the DDoS attack. After inserting flows with the action of discarding malicious packets (time bin 5), it was necessary to add new mitigation flows with the IP addresses of the new incoming attackers (time bin 6). Around time bin 8 the flows began decreasing, and traffic returned to normal as expected. In Fig. 9(b) it is possible to observe a peak in CPU usage around 90s. This peak is caused by the detection and mitigation module's activity for DDoS identification and implementation of the *load_balance* policy. From then on, the CPU was required to perform load balancing so that its use remained higher than usual throughout the

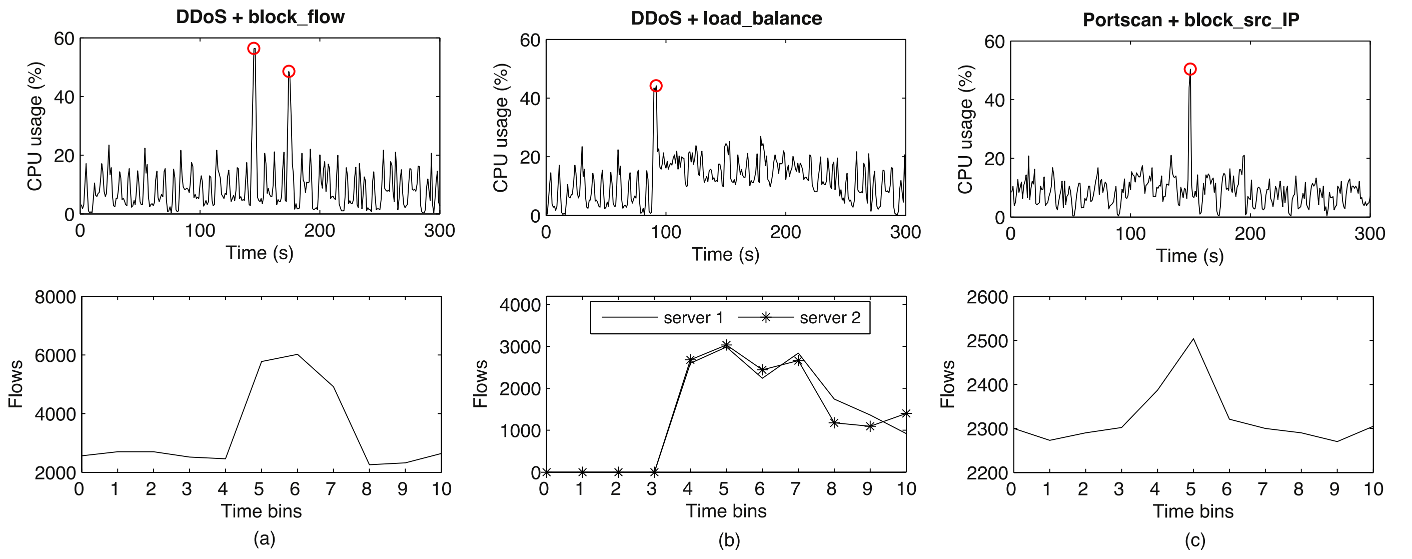


Fig. 9. System performance evaluated according to the type of threat and the mitigation policy. A time bin represents a 30 s time interval.

anomalous event. Finally, CPU usage in Fig. 9(c) depicts a peak when the portscan attack is encountered. As the *block_src_ip* policy only needs to discover a misbehaving source IP cut off its communication by inserting a single flow-entry in the switches, the peak lasts less than those observed in the detection of DDoS attacks. Accordingly, the flows in S5 decrease drastically as a consequence of the mitigation action.

5. Conclusion

This work presented an ecosystem designed to detect and mitigate network threats in an SDN environment. The system autonomously monitors the traffic and implements countermeasures on affected devices to maintain the availability of the network's services. To this end, the system employs a multi-feature traffic analysis to profile normal network usage. Thereafter, the resulting normal profile is used to identify unusual traffic patterns, and a mitigation policy is chosen according to the recognized anomalies.

The ecosystem was evaluated using a testbed simulating DDoS and port scan attacks. Compared to other anomaly detection approaches, our detection mechanism proved outstanding in terms of accuracy and low false-positive rate. The system locates interfaces that attackers have compromised. The mitigation module uses this information to begin mitigation routines at the affected switches and, therefore, restores proper network functionality. Regarding performance, we evaluated our proposed system's performance concerning execution time and system resources usage (controller's CPU and switches flow table size). This analysis demonstrated that the detection strategy did not vary significantly concerning the type of recognized anomaly as there is no need to retrain if we apply the MLR approach. In contrast, the more intense the attack, the more time was spent to execute the mitigation and preprocessing routines. However, even under high attack intensity, the modules remain operative, and they perform in milliseconds. By evaluating the controller's CPU usage and the number of active flows in the switches, it was possible to observe the effectiveness of the policies for preventing the attacks from spreading. The results indicated that the proposed mechanism could quickly start the attack detection and prescribe countermeasures to the switches, although it requires some time to recognize a DDoS attack.

The ecosystem presents a modular design, as statistic collection, anomaly detection, mitigation, and report tasks are decoupled and can conveniently be adapted to address new security issues. In this manner, in future work, we intend to study and incorporate into our ecosystem for SDN networks new anomaly detection techniques as well as corresponding mitigation policies. We will also extend this work to consider zero-day attacks that may compromise proper SDN operations.

Acknowledgment

This work was supported by the National Council for Scientific and Technological Development (CNPq) of Brazil under Grant of Project 308348/2016-8 and 304066/2015-0.

References

Ahmed, M., Mahmood, A. N., & Hu, J. (2016). A survey of network anomaly detection techniques. *Journal of Network and Computer Applications, 60*, 19–31. doi:10.1016/j.jnca.2015.11.016.

Aleroud, A., & Alsmadi, I. (2016). Identifying dos attacks on software defined networks: A relation context approach. In *Proceedings of the 2016 IEEE/IFIP network operations and management symposium, Noms 2016* (pp. 853–857). doi:10.1109/NOMS.2016.7502914.

Amaral, A. A., de Souza Mendes, L., Zarpelão, B. B., & Proença, M. L., Jr. (2017). Deep IP flow inspection to detect beyond network anomalies. *Computer Communications, 98*, 80–96. doi:10.1016/j.comcom.2016.12.007.

de Assis, M. V. O., Hamamoto, A. H., Abrão, T., & Proença, M. L. (2017). A game theoretical based system using holt-winters and genetic algorithm with fuzzy logic for DoS/DDoS mitigation on SDN networks. *IEEE Access, 5*, 9485–9496.

Bhuyan, M. H., Bhattacharyya, D. K., & Kalita, J. K. (2014). Network anomaly detection: Methods, systems and tools. *IEEE Communications Surveys Tutorials, 16*(1), 303–336. doi:10.1109/SURV.2013.052213.00046.

Braga, R., Mota, E., & Passito, A. (2010). Lightweight DDoS flooding attack detection using NOX/openflow. In *Proceedings of the 2010 IEEE 35th conference on local computer networks (LCN)* (pp. 408–415). doi:10.1109/LCN.2010.5735752.

Carvalho, L. F., Barbon, S., Jr., Mendes, L. S., & Proença, M. L., Jr. (2016). Unsupervised learning clustering and self-organized agents applied to help network management. *Expert Systems with Applications, 54*, 29–47.

Chen, K., Junuthula, A. R., Siddhrau, I. K., Xu, Y., & Chao, H. J. (2016). Sdnshield: Towards more comprehensive defense against DDoS attacks on SDN control plane. In *Proceedings of the 2016 IEEE conference on communications and network security (CNS)* (pp. 28–36). doi:10.1109/CNS.2016.7860467.

Cui, Y., Yan, L., Li, S., Xing, H., Pan, W., Zhu, J., et al. (2016). Sd-anti-DDoS: Fast and efficient DDoS defense in software-defined networks. *Journal of Network and Computer Applications, 68*, 65–79. doi:10.1016/j.jnca.2016.04.005.

Fernandes, G., Jr., Carvalho, L. F., Rodrigues, J. J., & Proença, M. L., Jr. (2016). Network anomaly detection using IP flows with principal component analysis and ant colony optimization. *Journal of Network and Computer Applications, 64*, 1–11. doi:10.1016/j.jnca.2015.11.024.

Giotis, K., Argyropoulos, C., Androulidakis, G., Kalogeras, D., & Maglaris, V. (2014). Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments. *Computer Networks, 62*, 122–136. doi:10.1016/j.bjp.2013.10.014.

Ha, T., Kim, S., An, N., Narantuya, J., Jeong, C., Kim, J., & Lim, H. (2016). Suspicious traffic sampling for intrusion detection in software-defined networks. *Computer Networks, 109*(Part 2), 172–182. doi:10.1016/j.comnet.2016.05.019.

Hamamoto, A. H., Carvalho, L. F., Sampaio, L. D. H., Abrão, T., & Proença, M. L. (2018). Network anomaly detection system using genetic algorithm and fuzzy logic. *Expert Systems with Applications, 92*(Suppl C), 390–402. doi:10.1016/j.eswa.2017.09.013.

Han, W., & Lei, C. (2012). A survey on policy languages in network and security management. *Computer Networks, 56*(1), 477–489. doi:10.1016/j.comnet.2011.09.014.

Hommes, S., State, R., & Engel, T. (2014). Implications and detection of dos attacks in openflow-based networks. In *Proceedings of the 2014 IEEE global communications conference (globecom)* (pp. 537–543). doi:10.1109/GLOCOM.2014.7036863.

hping3 (2006). http://www.hping.org/.

Jammal, M., Singh, T., Shami, A., Asal, R., & Li, Y. (2014). Software defined networking: State of the art and research challenges. *Computer Networks, 72*, 74–98. doi:10.1016/j.comnet.2014.07.004.

Jiang, J., & Papavassiliou, S. (2006). Enhancing network traffic prediction and anomaly detection via statistical network traffic separation and combination strategies. *Computer Communications, 29*(10), 1627–1638. doi:10.1016/j.comcom.2005.07.030. Monitoring and measurements of IP networks

Joldzic, O., Djuric, Z., & Vuletic, P. (2016). A transparent and scalable anomaly-based dos detection method. *Computer Networks, 104*, 27–42. doi:10.1016/j.comnet.2016.05.004.

Kreutz, D., Ramos, F. M., & Verissimo, P. (2013). Towards secure and dependable software-defined networks. In *Proceedings of the second ACM SIGCOMM workshop on hot topics in software defined networking HotSDN' 13* (pp. 55–60). New York, NY, USA: ACM. doi:10.1145/2491185.2491199.

Lantz, B., Heller, B., & McKeown, N. (2010). A network in a laptop: Rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM workshop on hot topics in networks Hotnets-IX* (pp. 19:1–19:6). New York, NY, USA: ACM. doi:10.1145/1868447.1868466.

Li, W., Meng, W., & Kwok, L. F. (2016). A survey on openflow-based software defined networks: Security challenges and countermeasures. *Journal of Network and Computer Applications, 68*(Suppl C), 126–139. doi:10.1016/j.jnca.2016.04.011.

Lin, S.-C., Wang, P., & Luo, M. (2016). Control traffic balancing in software defined networks. *Computer Networks, 106*(Suppl C), 260–271. doi:10.1016/j.comnet.2015.08.004.

Masoudi, R., & Ghaffari, A. (2016). Software defined networks: A survey. *Journal of Network and Computer Applications, 67*(Suppl C), 1–25. doi:10.1016/j.jnca.2016.03.016.

Mousavi, S. M., & St-Hilaire, M. (2015). Early detection of DDoS attacks against SDN controllers. In *Proceedings of the 2015 international conference on computing, networking and communications (ICNC)* (pp. 77–81). doi:10.1109/ICCNC.2015.7069319.

Nunes, I., Schardong, F., & Schaeffer-Filho, A. (2017). Bdi2dos: An application using collaborating BDI agents to combat DDoS attacks. *Journal of Network and Computer Applications, 84*, 14–24. doi:10.1016/j.jnca.2017.01.035.

ONF (2009). Openflow switch specification 1.0.0. https://www.opennetworking.org/.

Pena, E. H., Carvalho, L. F., Barbon, S., Jr., Rodrigues, J. J., & Proença, M. L., Jr. (2017). Anomaly detection using the correlational paraconsistent machine with digital signatures of network segment. *Information Sciences, 420*(Suppl C), 313–328. doi:10.1016/j.ins.2017.08.074.

Phan, T. V., Toan, T. V., Tuyen, D. V., Huong, T. T., & Thanh, N. H. (2016). Openflowsia: An optimized protection scheme for software-defined networks from flooding attacks. In *Proceedings of the 2016 IEEE sixth international conference on communications and electronics (ICCE)* (pp. 13–18). doi:10.1109/CCE.2016.7562606.

Proença, M. L., Jr., Coppelmans, C., Bottoli, M., Alberti, A., & Mendes, L. S. (2004). In J. N. de Souza, P. Dini, & P. Lorenz (Eds.), *The hurst parameter for digital signature of network segment* (pp. 772–781)). Berlin, Heidelberg: Springer Berlin Heidelberg. Fortaleza, Brazil, August 1–6, 2004

Sahay, R., Blanc, G., Zhang, Z., Toumi, K., & Debar, H. (2017). Adaptive policy-driven attack mitigation in SDN. In *Proceedings of the 1st international workshop on security and dependability of multi-domain infrastructures XDOMO'17* (pp. 4:1–4:6). New York, NY, USA: ACM. doi:10.1145/3071064.3071068.

Sandhya, Sinha, Y., & Haribabu, K. (2017). A survey: Hybrid SDN. *Journal of Network and Computer Applications*. doi:10.1016/j.jnca.2017.10.003.

Scapy (2017). http://scapy.readthedocs.io.

Wang, B., Zheng, Y., Lou, W., & Hou, Y. T. (2015). DDoS attack protection in the era of cloud computing and software-defined networking. *Computer Networks, 81*, 308–319. doi:10.1016/j.comnet.2015.02.026.

Wang, Y. (2005). A multinomial logistic regression modeling approach for anomaly intrusion detection. *Computers & Security, 24*(8), 662–674. doi:10.1016/j.cose.2005.05.003.

Wang, Y., Zhang, Y., Singh, V., Lumezanu, C., & Jiang, G. (2013). Netfuse: Short-circuiting traffic surges in the cloud. In *Proceedings of the 2013 IEEE international conference on communications (ICC)* (pp. 3514–3518). doi:10.1109/ICC.2013.6655095.

Wei, L., & Fung, C. (2015). Flowranger: A request prioritizing algorithm for controller dos attacks in software defined networks. In *Proceedings of the 2015 IEEE international conference on communications (ICC)* (pp. 5254–5259). doi:10.1109/ICC.2015.7249158.

Xiao, P., Qu, W., Qi, H., & Li, Z. (2015). Detecting DDoS attacks against data center with correlation analysis. *Computer Communications, 67*, 66–74. doi:10.1016/j.comcom.2015.06.012.

Yuan, B., Zou, D., Yu, S., Jin, H., Qiang, W., & Shen, J. (2017). Defending against flow table overloading attack in software-defined networks. *IEEE Transactions on Services Computing, PP*(99) 1–1. doi:10.1109/TSC.2016.2602861.

Zaalouk, A., Khondoker, R., Marx, R., & Bayarou, K. (2014). Orchsec: An orchestrator-based architecture for enhancing network-security using network monitoring and SDN control functions. In *Proceedings of the 2014 IEEE network operations and management symposium (NOMS)* (pp. 1–9). doi:10.1109/NOMS.2014.6838409.