A Novel Anomaly Detection System to Assist Network Management in SDN Environment

Luiz F. Carvalho*, Gilberto Fernandes Jr.†, Joel J. P. C. Rodrigues†‡§, Leonardo S. Mendes*, Mario Lemes Proença Jr.¶
*School of Electrical and Computer Engineering, States University of Campinas, Brazil
†Instituto de Telecomunicações, Universidade da Beira Interior, Portugal
‡National Institute of Telecommunications (INATEL), Brazil
§University of Fortaleza (UNIFOR), Brazil
¶Computer Science Department, States University of Londrina, Brazil
{luizfcarvalhoo, gil.fernandes6}@gmail.com, joeljr@ieee.org, lmendes@decom.fee.unicamp.br, proenca@uel.br

***Abstract*—Software-Defined Networking (SDN) emerges as a recent paradigm that grants a holistic network visibility and flexible network programmability, facilitating rapid innovation of protocol and services. Although SDN provides greater control over traffic flow than ever before, it also introduced new challenges and issues to be addressed with its management. In that light, the security and reliability of SDN have been neglected subjects. This paper presents a system designed to proactively monitor network traffic and autonomously detect anomalies which may impair the proper network functioning. In addition to the identification of anomalous events, the proposed approach also provides routines that allow the mitigation of the anomalies effects.**

I. Introduction

Although the convenience provided by communication networks continually stimulate the demand for increasingly improved services, the Internet's ability to handle large traffic patterns resulting from new types of emerging services (server virtualization, cloud computing, big data) is fast becoming insufficient [1]. The issue to address is that networks were designed the same way for decades, but the types of applications demanded by users have changed extensively. This made networks become complex systems throughout their evolution. Currently, networks must be fast and flexible to support the needs of its customers, highly dynamic application deployment environments with hundreds of virtual machines created and dissolved every day, and a set of changing applications and equipment [2].

In this context, Software-Defined Networking (SDN) raises as a promising alternative architecture, enabling scalability and unprecedented flexibility in the configuration and deployment of services. The greatest feature of this recent paradigm is the dissociation of the data and network control planes, and the transfer of the latter to a centralized controller application. With this separation, various network devices may share the same controller. A major advantage is that if there is the need for policy change or the establishment of service quality, it can be done through control plane programming, which uses a more intuitive high-level language than device configuration commands. Indeed, the administrator can modify network forward rules, prioritizing or even blocking specific types of traffic.

In spite of SDN networks providing greater control over traffic flow than ever before, due to its dynamic nature, they also introduced new challenges and issues to be addressed during their management [3]. From the same perspective, the security of SDN have been a neglected subject so far. In that light, the main purpose of this paper is to present a system that assists the administrator in managing SDN networks, performing real-time anomaly detection and mitigation. Such a system is able to perform constant traffic monitoring using traffic analysis of packet flows.

The proposal is divided into four complementary stages. The first corresponds to the collection and storage of flows through OpenFlow protocol, which has been widely used to collect network flow statistics passively or measure actively latency through the packets injection traffic analysis. The second phase profiles the normal traffic behavior in order to create a baseline or digital signature of the analyzed segments. Afterward, current traffic is compared with the previously established traffic profile in order to identify suspicious traffic events that differ from the expected behavior. 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.

The remainder of this paper is organized as follows. Section II shows the related work. In Section III, we present the system design principles. Section IV describes the results and performance of the system. Finally, Section V concludes the paper.

II. Related Work

Despite the extensive work in SDN environments with respect to monitoring, load balancing, and routing, few approaches to detecting anomalies have been proposed. Early work on establishing a flow-based anomaly detection approach using SDN concepts was reported by [4]. The authors used Self-Organizing Maps (SOM) with the topological neighborhood described by a Gaussian function to classify each packet

978-1-4673-8999-0/17/$31.00 ©2017 IEEE

as benign or abnormal. Although promising results have been achieved, their work is restricted to DDoS attacks detection. Another attempt focusing on traffic anomaly detection on SDN environments is presented in [5]. Its centralized monitoring application uses variations in the logical topology of the network to detect DoS attacks and a basic mitigation rule to react against the huge traffic burst. Likewise, Wang *et al.* [6] 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 the network devices and network controller as an additional layer. To detect overloading behavior, a multi-dimensional flow aggregation algorithm is employed that automatically finds the flow clusters that are suspicious. Other researchers like [7] also focus on exploiting the benefits of SDN to defend the same kind of anomaly. However, their target victims still reside in the traditional network environment, which makes their solutions unsuitable for the new network paradigm.

Still focusing on detection and mitigation of DDoS, Ha *et al.* [8] 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). Joldzic *et al.* [9] proposed 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 forward the generated parts to the intermediate layer. The second layer is composed of several devices called processors, which perform the detection of the attack. At the third layer, the traffic generated by the processors are aggregated and sent to the interior of the network through an OpenFlow switch.

Aleroud and Alsmadi [10] proposed a model for detecting DDoS attacks that affect the communication between the switches and the SDN controller. Traffic is aggregated and classified according to presets arranged in a graph and attack detection is performed by the K-NN classification technique. Similarly, in [11] an effective detection approach based on traffic classification with correlation analysis (CKNN) was proposed to detect the attack. Although the detection rate of the events already known by the approaches is high, the methods proposed by the authors become incapable of recognizing variations of the same attack, since signature-based detection is used. For this same reason, the techniques that are part of the proposals should be trained with a large labeled dataset, which is not always possible to obtain.

In this paper, we focus specifically on SDN environment, which the controller is able to collect and analyze periodic statistics reports from switches regarding the current status of traffic through the OpenFlow protocol. Our approach uses this information to detect and classify a variety of anomalies in addition to choosing the right strategy for their mitigation. Also, the proposed system provides reports to the network administrator regarding the details of detected anomalies and efficiency of countermeasures taken against these issues.

III. System Design Principles

The design of the presented system on anomaly detection and mitigation in SDN-specific environments are based on the following properties:

- *Autonomic approach:* ensures greater accuracy and reduction of errors arising from human intervention.
- *Modular design:* Although tasks such as traffic statistics collection, anomaly detection, mitigation of the attacks and report generating are complementary in our approach, they are designed to be decoupled.
- *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 approaches that analyze the entire aggregate network traffic, our proposal 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., their normal profile. In case 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.

Recently proposed Software-Defined Networks control underlying network devices via an external control plane. This control plane consists of three parts: a network operating system, a protocol for packet forwarding management (commonly OpenFlow protocol) and network applications that run on top of the network operating system [4]. In our approach, data collection, anomaly detection, mitigation and reporting routines are built upon the controller as network applications. Each of them is detailed in the following subsections.

A. Statistic Collection Module

In this paper, the information statistics gathering is made 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 [4]. In addition, it uses the concept of flows to identify network traffic based on pre-defined rules, programmed in the SDN controller software. Flows are defined as a group of packets with one-way transmissions sessions that share features as transport protocol, addresses, and both source and destination ports. Since flows are identified by common characteristics, they often can be related to an application, network device or user. As a result, administrators or automated management systems may define how traffic should flow through network devices.

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. The data collected on each switch are processed separately from information acquired from another. Thus, the primary challenge of identify which OpenFlow switch is being affected by the anomaly is straightforward.

Several studies in the literature explore detecting anomalies using traffic analysis at fixed period of five-minute interval [12]. However, due to the continuous increase in transmission rates, it becomes impractical. For example, a network that operates at 10 Gbps may be Terabits of information compromised during the five-minute interval. Thus, we adopted a time window of 30 seconds to request statistics from switches and send them to the anomaly detection module ensuring that countermeasures are triggered in near real-time.

In our approach, collected per-flow statistics are a compound of six features: i) Received packets that measure the amount of packets matching related with the flow they belong to; ii) Received bits which represents the total of bits received by a flow; iii) Source IP address; iv) Destination IP address; v) Source port and vi) Destination port. The last four attributes are qualitative metrics and the benefits of its use are twofold. Firstly, they provide information to recognize devices and applications which are operating suspiciously. In addition, they become sensitive indicators of traffic behavior change 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 amount collected in each analyzed time window and for IP address and ports features we enable their use in traffic analyzing adopting the Shannon Entropy. For its calculation, it is necessary to create a histogram from samples of each traffic measurement. Given the attribute $X = \{n_1, \ldots, n_i, \ldots, n_N\}$, in which n_i presents the occurrences number of sample i, the entropy H for X is defined as:

$$H(X) = -\sum_{i=1}^{N} \left(\frac{n_i}{S}\right) \log_2 \left(\frac{n_i}{S}\right), \tag{1}$$

where $S = \sum_{i=1}^{N} n_i$ is the total of all occurrences present in the histogram. Applying this technique, we do not only get the benefit of summarizing the information related to IP addresses and ports over a period of time, as to represent quantitatively the dispersion or concentration degree of these attributes. For example, the distribution of features may become more dispersed with respect to source address, such as a DDoS attack or when ports are scanned in order to find vulnerabilities. Furthermore, when a large amount of connections is established by a particular source, the IP address distribution gets more concentrated.

B. Anomaly Detection Module

Detection of anomalous behavior in network traffic is an important resource for network management. Among different methodologies, the creation of network behavior profile stands out for providing dynamic traffic monitoring and detection. As a result, more attention has to be drawn to these methods since they are able to identify even novel types of anomalies. In [13], it has observed that traffic is currently composed of cycles consisting of bursts with particular characteristics of network usage. Therefore, our approach recognizes these behaviors and their characteristics in order to create a normal traffic profile. Then, using this knowledge, the network is continuously monitored to recognize time windows where current traffic seems to be unusual.

TABLE I
EXAMPLES OF ANOMALIES AND THEIR EFFECTS ON TRAFFIC FEATURES.

Anomaly	Traffic Features Affected
Worms	Source address, destination address and destination port
DDoS	Source address, destination address, destination port and some volume features
Port Scan	Destination address, destination port and source address

As previously mentioned, the anomaly detection module is composed of two levels. In the first one, a traffic characterization of each OpenFlow switch is performed in order to extract usual events presented in the historical traffic. This process is accomplished for all of six collected features and its outcome is a normal traffic profile represented by matrix $\mathbf{P}_{n \times a}$, where n is the total time window of a day and a is the data dimensionality, or the number of traffic features. Therefore, p_{ij} indicates the expected value of j-th feature at time window i. It is important to emphasize that monitoring and characterization of traffic are concurrent since measurements of recent past time windows are automatically embedded in the historical traffic.

A suspicious event is detected by deviation of monitored traffic in relation to the normally expected profile. Although alarms are generated for each of six analyzed attributes, an anomaly is solely considered by a general alarm, which may be triggered when observing an anomalous traffic showed in Table I or an unexpected behavior correlates at least four flow features at current time window. The table lists anomalies commonly found in a backbone traffic and their effects on flow measurements. If no correspondence between the behavior generated by the anomaly and signatures recognized by the system exists, it is considered unknown and its signature is stored for later detections.

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

Although our approach can detect the anomalies shown in Table I, there are some legitimate network abnormalities which affect the traffic features in the same way that a malicious anomaly does. In order to avoid interruption of benign traffic events, we implemented a whitelist to store IP addresses and ports that their unusual behavior might be considered as anomaly-free traffic. Thus, before to apply mitigation routines, it is checking whether IP addresses and ports related to anomaly are in the whitelist. The whitelist can be maintained by an automated procedure or set manually by the network manager. In this paper, we implemented the latter for simplicity reasons.

C. Mitigation Module

Our approach classifies network behavior in four different status: *normal*, *shifting pattern*, *anomalous* and *mitigation in progress*. A *shifting pattern* occurs by fluctuations in the traffic measurements, which results from small variations of legitimate network usage. On the other hand, it can be the beginning of a malicious occurrence that leads to the anomalous condition. *Mitigation in progress* status happens when actions are employed to contain an anomaly, ensuring the network returns to its normal state.

Once the detection module identifies an anomaly, it passes on relevant information to the mitigation module about the detected event, such as the type, timestamp, IP address of the attacker and ports involved in the malicious communication. An exception occurs during a DDoS attack, in which only the target host and service are identified. Due to numerous attackers, the implementation of mitigating actions to deal with each of them would be infeasible [14].

In short, our proposal aims to apply policy rules to eliminate the traffic that harms network normal operation. For this purpose, flow entries containing the IP address and port learned in the anomaly detection module are inserted into forwarding table 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* packets fields [15]. We attach *forward* action to each benign flow entry while *drop* action is used in order to block malicious traffic. After the countermeasure is designed, our approach pushes the policy to the switch through network controller. This can be accomplished using the OpenFlow protocol, which acts directly in the forwarding logic of 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 work as long as the anomaly is happening.

D. Reporting Module

A weakness of the profile-based detection is that anomalies can be slowly introduced in the normal pattern until they become a legitimate event. The presented system offers the administrator relevant information which helps with the solution of such issues. This information is arranged in reports, or diagnoses. These demonstrate the use of network resources at the moment that an anomaly is detected. For instance, the provided diagnostics assist in determining if an unknown anomalous event is expected variations of normal behavior or a security threat. Besides that, these reports help the network administrator to make correct decisions to reduce the impact in the services offered to end users. In this manner, if these anomalous events are interrupted, they are not included in the historical dataset and consequently will not compose the normal traffic profile.

IV. Results and Analysis

To verify the performance of the presented solution, we have implemented it in the widely-used Mininet [16], an emulator which creates a network of virtual hosts, switches, controllers, and links. Codes developed on Mininet can be moved to a real environment with minimal changes for real-world deployment. As can be seen in Fig. 1, a tree-like network of depth two is emulated. All switches are interconnected by the root switch and each subnet is composed of forty hosts. In order to conduct our experiments, Open vSwitch was used for network switches. It is a software that can handle the required traffic and supports the OpenFlow protocol. For our statistic collection, anomaly detection, and mitigation tasks we implemented all the necessary algorithms as POX controller applications written in Python language.

The methodology assumed in this paper creates a normal network profile for each analyzed day based on the history of its previous weeks. Thus, we have employed Scapy [17], a programmable tool that sends packets sequences with arbitrary protocol field values (e.g. IP address, ports and content) to generate four days of historical traffic. Our test scenario was built to mimic real environments, being as realistic and reliable as possible by taking into account suggestions proposed by [4]. Thus, our analysis included mixing different types of legitimate traffic and varying attack traffic parameters. Benign traffic during tests combines about 85% of TCP traffic, 10% of UDP, and 5% of ICMP.

After ensuring that all collected traffic features are represented by quantitative values, the extracting normal behavior begins. In this paper we build upon one of our previous techniques [18] for traffic profiling by using it to provide a basis for anomaly detection. Primarily, the analyzed attributes are provided on a six-tuple composed of bits and packets counters followed by entropy calculated for source IP, destination IP, source and destination port, respectively. Thereafter, we use a

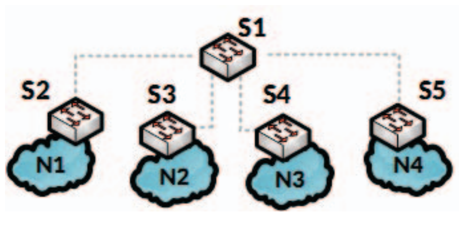

Fig. 1. Tree-like topology emulated on Mininet.

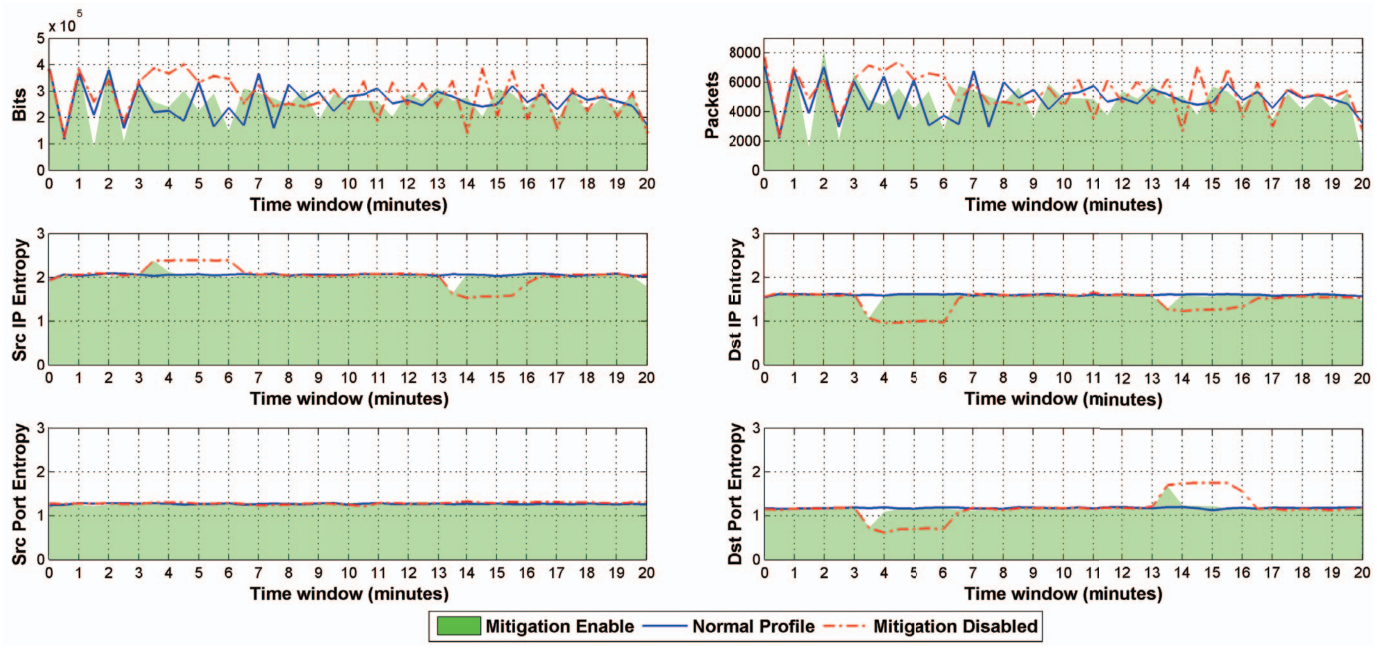

Fig. 2. Traffic features values during DDoS and Portscan attacks with and without the mitigation mechanism.

data clustering approach in order to discover sensible organization of each historical traffic time window in order to identify and quantify similarities between them. The clustering process is conducted by Ant Colony Optimization for Digital Signature (ACODS), a modification of traditional ACO metaheuristic. The advantage of using the ACODS is that the clustering process is an unsupervised learning technique, allowing the system to work in an autonomic manner. In addition, it enables the construction of solutions not given by local optimal, which is the existing problem in some clustering algorithms.

In order to find traffic behaviors that differ from the normal profile, a similarity measure should be adopted. The Euclidean distance between each point of the same index has been widely used in time series for this purpose. However, this metric is not suitable for identifying shifts in the data sequence. Believing that normal traffic behavior can suffer such displacements due to the changes in the schedule of users' activities, ACODS implements Adaptive Dynamic Time Warping (ADTW) to fit these situations. This method is a modification of Dynamic Time Warping (DTW) pattern matching technique widely used to find an optimal alignment between two time series.

Aiming to test the system effectiveness, synthetic anomalies flow traces were generated to mimic certain attacks behaviors. The first trace is a DDoS attack composed of multi-source generating requests to a single destination. The second anomaly corresponds to port scan attack, where a source sends packets with the SYN flag enabled to different ports of the destination host, aiming to receive the confirmation whether they are operative. Both attacks was directed to hosts of N4 in the evaluated topology.

Fig. 2 represents the six traffic attributes monitoring over a 20-minute interval related to switch S5. As seen, the anomalous traces were blended with background traffic. When the DDoS attack (at minute 3) and ports scan (at minute 13) were triggered, the monitored traffic differs significantly from the blue line, which depicts the normal profile learned by ACODS. The green area and the red line describe the collected data when the attack mitigation module is used or not, respectively. It is noticeable that once the attack is identified and mitigated, the features values return to the expected range.

As previously mentioned, during the second level of traffic monitoring the system performs a fine-grained analysis about suspicious communication. Fig. 3 shows key information about the port scan attack in the time window it was recognized. In this situation, a few range of ports (1 to 3000) were scanned by the fictitious IP 200.200.200.200 with different source ports, characterizing an attack with low network traffic impact. This host have sent packets to various ports of destination

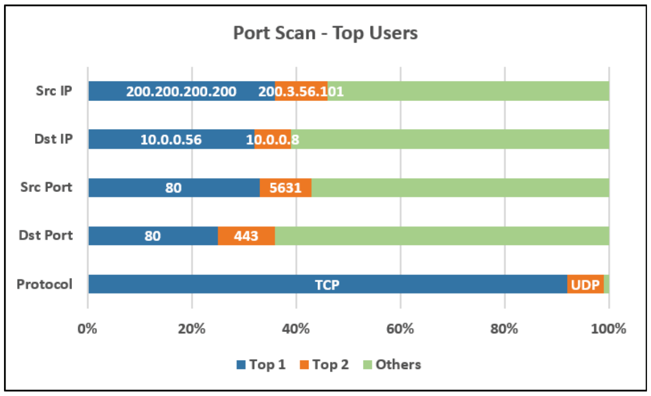

Fig. 3. Key information of port scan attack.

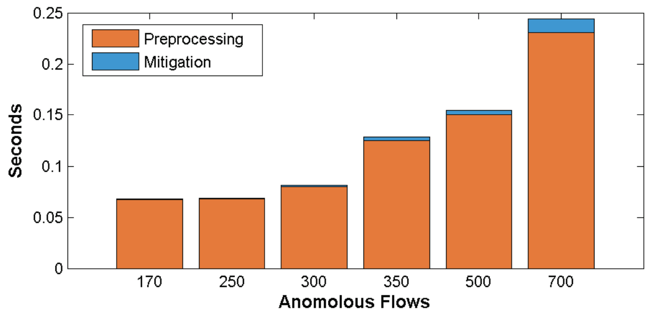

Fig. 4. Execution time according to DDoS traffic load.

host 10.0.0.56 using TCP protocol. Our approach identify this behavior and a mitigation routine is triggered, providing a self-healing characteristic to the network. Suspending the communication destined to the host under attack is a simple way to mitigate a DDoS attack considering that all packets to the victim will be temporarily awarded the drop action. Port scan attack mitigation is performed in a similar way to DDoS. In this case, the only difference is that we cut off the host attacking a set of destination ports.

To evaluate the system performance, Fig. 4 shows the amount of spent time to execute the system's routines according to different DDoS traffic loads. We define preprocessing as statistic collection and entropy calculation. Similarly, mitigation time corresponds to time spent to assign source and destination of detected anomaly and push policies to the switch through network controller. It is important to note that the communication time between the switches and the controller is not taken into account. Even under varying attack intensities, the modules remain operative and they perform in milliseconds. The increase in time spent in each activity execution is due to the number of flows treated by the modules, not only the anomalous flows but also the benign ones, which during the experiments they reached peaks of approximately two thousand flows sent to subnet N4.

V. Conclusion

In this paper, we investigate the problem of anomaly detection in SDN environment and present an autonomous detection system, which performs traffic characterization of six flows attributes to provide real-time detection and mitigation. Our approach is designed as a modular structure since data gathering, anomaly detection and anomaly mitigation functions are decoupled and can conveniently be adapted to address future security issues. We tested the system using DDoS and port scan attacks and the results confirm that the system is capable of identifying the victim or attackers in order to mitigate malicious traffic. Thus, we conclude that the proposed system contributes to the network management promoting their reliability and availability of the services offered by it.

In future work, we plan to use a real SDN testbed to evaluate our proposed solution. In addition, we intend to allow communication between inter-domain controllers to prevent anomaly spreading.

Acknowledgment

This work has been partially supported by the National Council for Scientific and Technological Development (CNPq) of Brazil under Grant of Project 308348/2016-8 and due to the concession a scholarship under process 249794/2013-6. Also, this work has been partially supported by the *Instituto de Telecomunicações*, Next Generation Networks and Applications Group (NetGNA), Portugal, by National Funding from the FCT - *Fundação para a Ciência e a Tecnologia* through the UID/EEA/50008/2013 Project, and by Finep, with resources from Funttel, Grant No. 01.14.0231.00, under the *Centro de Referência em Radiocomunicações* - CRR project of the *Instituto Nacional de Telecomunicações* (Inatel), Brazil.

References

[1] A. Hakiri, A. Gokhale, P. Berthou, D. C. Schmidt, and T. Gayraud, "Software-defined networking: Challenges and research opportunities for future internet," *Computer Networks*, vol. 75, pp. 453 – 471, 2014.

[2] M. Jammal, T. Singh, A. Shami, R. Asal, and Y. Li, "Software defined networking: State of the art and research challenges," *Computer Networks*, vol. 72, pp. 74 – 98, 2014.

[3] A. Juhola, T. Ahola, and K. Ahola, "Adaptive risk management with ontology linked evidential statistics and sdn," in *Proceedings of the 2014 European Conference on Software Architecture Workshops (ECSAW)*, 2014, pp. 2:1–2:7.

[4] R. Braga, E. Mota, and A. Passito, "Lightweight ddos flooding attack detection using nox/openflow," in *2010 IEEE 35th Conference on Local Computer Networks (LCN)*, Oct 2010, pp. 408–415.

[5] S. Hommes, R. State, and T. Engel, "Implications and detection of dos attacks in openflow-based networks," in *2014 IEEE Global Communications Conference (GLOBECOM)*, Dec 2014, pp. 537–543.

[6] Y. Wang, Y. Zhang, V. Singh, C. Lumezanu, and G. Jiang, "Netfuse: Short-circuiting traffic surges in the cloud," in *2013 IEEE International Conference on Communications (ICC)*, June 2013, pp. 3514–3518.

[7] D. Kreutz, F. M. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '13. New York, NY, USA: ACM, 2013, pp. 55–60.

[8] T. Ha, S. Kim, N. An, J. Narantuya, C. Jeong, J. Kim, and H. Lim, "Suspicious traffic sampling for intrusion detection in software-defined networks," *Computer Networks*, vol. 109, Part 2, pp. 172 – 182, 2016, traffic and Performance in the Big Data Era.

[9] O. Joldzic, Z. Djuric, and P. Vuletic, "A transparent and scalable anomaly-based dos detection method," *Computer Networks*, vol. 104, pp. 27 – 42, 2016.

[10] A. Aleroud and I. Alsmadi, "Identifying dos attacks on software defined networks: A relation context approach," in *2016 IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 2016, pp. 853–857.

[11] P. Xiao, W. Qu, H. Qi, and Z. Li, "Detecting {DDoS} attacks against data center with correlation analysis," *Computer Communications*, vol. 67, pp. 66 – 74, 2015.

[12] A. A. Amaral, L. de Souza Mendes, B. B. Zarpelão, and M. L. P. Jr., "Deep {IP} flow inspection to detect beyond network anomalies," *Computer Communications*, vol. 98, pp. 80 – 96, 2017.

[13] G. F. Jr., J. J. Rodrigues, and M. L. P. Jr., "Autonomous profile-based anomaly detection system using principal component analysis and flow analysis," *Applied Soft Computing*, vol. 34, pp. 513 – 525, 2015.

[14] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining openflow and sflow for an effective and scalable anomaly detection and mitigation mechanism on {SDN} environments," *Computer Networks*, vol. 62, pp. 122 – 136, 2014.

[15] "The openflow specification version 1.0.0." [Online]. Available: http://archive.openflow.org/

[16] "Mininet." [Online]. Available: http://mininet.org/

[17] "Scapy." [Online]. Available: http://www.secdev.org/projects/scapy/

[18] L. F. Carvalho, S. B. Jr., L. de Souza Mendes, and M. L. P. Jr., "Unsupervised learning clustering and self-organized agents applied to help network management," *Expert Systems with Applications*, vol. 54, pp. 29 – 47, 2016.