ATLANTIC: A Framework for Anomaly Traffic Detection, Classification, and Mitigation in SDN

Anderson Santos da Silva, Juliano Araujo Wickboldt, Lisandro Zambenedetti Granville, Alberto Schaeffer-Filho
Institute of Informatics, Federal University of Rio Grande do Sul, Porto Alegre, Brazil
Email: {assilva, jwickboldt, granville, alberto}@inf.ufrgs.br

***Abstract*—Anomaly traffic detection and classification mechanisms need to be flexible and easy to manage in order to detect the ever growing spectrum of anomalies. Detection and classification are difficult tasks because of several reasons, including the need to obtain an accurate and comprehensive view of the network, the ability to detect the occurrence of new attack types, and the need to deal with misclassification. In this paper, we argue that Software-Defined Networking (SDN) form propitious environments for the design and implementation of more robust and extensible anomaly classification schemes. Different than other approaches from the literature, which individually tackle either anomaly detection or classification or mitigation, we present a management framework to perform these tasks jointly. Our proposed framework is called ATLANTIC and it combines the use of *information theory* to calculate deviations in the entropy of flow tables and a range of *machine learning* algorithms to classify traffic flows. As a result, ATLANTIC is a flexible framework capable of categorizing traffic anomalies and using the information collected to handle each traffic profile in a specific manner, *e.g.,* blocking malicious flows.**

***Index Terms*—Software-Defined Networking, Network Management, OpenFlow, Anomaly detection**

I. Introduction

Computer networks must be resilient and properly deliver the communication services expected by their users [1]. The detection of an ever increasing number of anomalies in network traffic is the key task to achieve resilience. Anomaly detection in traditional computer networks is difficult to achieve because the points of observation are spread along distributed forwarding devices. With the advent of Software-Defined Networking (SDN) [2] [3], however, anomaly detection can be performed at the logically centralized spot created by the SDN controller. SDN in general allows building more reliable, extensible, and manageable networks where new network functions can be more easily deployed. However, SDN-based networks are not free of abnormal traffic that can affect the resilience of the network. Still, SDN can facilitate the design of anomaly detection and traffic classifications systems because of several reasons: (*i*) SDN offers a more comprehensive view of the network, (*ii*) SDN supports the easy collection of flow statistics, and (*iii*) SDN includes a dedicated management plane to coordinate dynamic reconfiguration actions.

To the best of our knowledge, there is no framework capable of managing anomaly detection, classification and mitigation in a coordinated manner in SDN environments. We advocate that such a framework should perform these tasks jointly, be fully extensible to accommodate different types of anomalies, and rely on modular software abstractions. To address these issues, in this paper, we introduce the ATLANTIC (*Anomaly deTection and machine LeArNing Traffic classifICation for software-defined networking*) framework for detection, classification, and mitigation of traffic anomalies in SDN-based networks. Anomaly detection and classification are performed in two complementary phases: (*i*) a *lightweight phase*, in which low computation cost methods are executed more frequently to quickly spot potentially malicious flows, and (*ii*) a *heavyweight phase*, where such flows are analyzed and classified according to their abnormal behavior. To instantiate our framework, we employ an information theory approach based on entropy analysis [5] in the *lightweight phase*, whereas in the *heavyweight phase* a machine learning algorithm based on Support Vector Machine (SVM) [6] is used to leverage historical knowledge about past anomalies and to classify the abnormal traffic.

Our main contributions are: (*i*) a strategy that obtains global network information without additional costs to network administrators, such as additional sensors; (*ii*) an architecture to combine several types of anomaly detection, classification, and mitigation techniques in a flexible manner, while avoiding high resource utilization; (*iii*) a publicly available application of how SDN can provide sophisticated software-based management solutions (represented by the lightweight and heavyweight phases) to tackle legacy network problems, such as managing classification techniques. We have developed a prototype system as a proof-of-concept. Our prototype has been implemented in Python and is publicly available in GitHub[1]. We evaluate the instantiation of our ATLANTIC framework to manage an SDN-based environment consisting of 11 switches organized according to the topology of the Federal University of Rio Grande do Sul campus network. In our experimental evaluation, we observed performance, accuracy, and overhead of ATLANTIC, considering distributed denial of service (DDoS) and port scanning attacks.

The remaining of this paper is organized as follows. In Section II, we present background and review related work. In Section III, we introduce ATLANTIC. In Section IV, we present our evaluation and associated results, including a performance analysis of the framework. In Section V, we

978-1-5090-0223-8/16/$31.00 © 2016 IEEE

[1]https://github.com/AndersonSanSilva/ATLANTIC

conclude this paper presenting final remarks and future work.

II. Background and Related Work

Anomaly detection and traffic classification in traditional networks are research areas that have been intensively investigated for years [7] [8] [9]. In SDN-based networks, however, the amount of work on this subject is still much less prominent. SDN-based as well as traditional networks are susceptible to abnormal traffic that can harm network operations and management, *e.g.,* Distributed Denial of Service (DDoS) attacks [10], [11]. As such, abnormal traffic must be detected, classified, and mitigated. Before introducing the ATLANTIC framework, we first present in this section background information on the key anomaly detection techniques and the related work.

Because of the emergence of new traffic behaviors, strategies to detect and classify anomalous traffic are necessary so to protect the network against malicious attacks. In traditional networks, machine learning has been widely used for traffic classification and anomaly detection [12]. These techniques can be divided into two main classes: *supervised learning* and *unsupervised learning*. Supervised learning techniques, such as Naive Bayes [13] and Support Vector Machine (SVM) [6], are suitable for classifying data samples into a range of known attacks. However, supervised learning is unable to handle new types of attacks. SVM typically achieves high classification accuracy [12] and thus it is commonly chosen to compose anomaly detection systems. On the other hand, unsupervised learning techniques, such as K-means [8] and Expectation Maximization [12], are suitable for detecting new types of attacks. However, in general, they need human input to determine the classes of the sampled data, which is generally grouped by similarity. Despite the high accuracy and performance obtained with some techniques, machine learning algorithms tend to suffer from several limitations: *(i)* the difficulty of determining the best set of discriminators to classify flows [13]; *(ii)* the availability of labeled training data for classification [8] [14]; *(iii)* the trade-offs between different machine learning algorithms regarding accuracy and performance [14]; *(iv)* the sheer amount of traffic data that makes it difficult to handle and to promptly detect malicious activities [15] [10]; *(v)* the availability of a high amount of resources, such as management systems and middleboxes, to collect traffic information [16]. Further discussion on machine learning and its use for network traffic classification is presented by Nguyen and Armitage [12].

Further, techniques based on Information Theory have also been used in traditional networks for anomaly traffic detection [15] [17]. These techniques use probability and statistic theory to model the entropy, *i.e.*, the mean information present in some set of traffic features, to detect when disturbances occur in the network. In particular, entropy can be used to model a high-level view of the flows observed in the network, and enables the monitoring of distributions of flow features with reduced computational cost. By analyzing the entropy information within a time interval it is possible to detect deviations that indicate an anomaly. Past research efforts indicate that entropy is a suitable, low-cost, and accurate technique to monitor traffic behavior changes [5]. Moreover, the combination of entropy and machine learning can be used for traffic classification [18].

Recent research efforts have indicated that SDN is suitable for the implementation of sophisticated software solutions and that anomaly detection schemes can benefit from the SDN architecture [11] [19] [20]. However, different than these works, we propose a framework that jointly coordinates anomaly detection, classification and mitigation tasks. SDN can assist to overcome legacy challenges related to anomaly detection, classification and mitigation because its software abstractions enhance the network visibility and management [21]. More concretely, *(i)* the OpenFlow protocol can natively collect primitive traffic statistics about traffic flows; and *(ii)* the network controller can be aware of network topology, traffic profiles, and forwarding behavior without relying on middleboxes. Our work is encouraged by the lack of an integrated framework that combines and manages a large set of techniques related to detection, classification and mitigation of network anomalies.

III. A framework for anomaly detection, classification and mitigation in SDN

In this paper, we advocate that anomaly detection and traffic classification can take advantage of SDN/OpenFlow characteristics. To demonstrate this, we introduce ATLANTIC, an anomaly detection, classification and mitigation framework that allows an administrator to flexibly reconfigure the operation of its building-block components and algorithms. In this section, we discuss the requirements of our framework and explain the main principles behind it. In addition, we present ATLANTIC in details and describe its main components. In particular, we show an overview of our traffic classification process and discuss our proposed architecture.

A. Framework Requirements

A framework for anomaly detection and traffic classification should be capable of orchestrating several different modules, such as those responsible for traffic monitoring, classification, and mitigation. We argue that the required functionality for such a framework should be placed in the management plane of the SDN architecture, and take into account the following aspects:

- **Comprehensive view of the network -** To perform traffic monitoring and analysis, the framework must be able to retrieve detailed and unrestricted information about the network and traffic flows. As opposed to applications sitting in the application plane of SDN – which make use of the Northbound API to request network resources to the controller – our framework uses the Management Interface to gain access to information about flows from all applications, and uses this information to manage traffic anomalies.
- **Human intervention -** The network administrator must be able to interact and monitor the operation of the

anomaly detection and traffic classification framework, observing logs and reconfiguring its operation whenever necessary. For example, an administrator might update parameters or replace some component functionality to increase performance or accuracy of classification.

- **Flexible network configuration -** Several types of configurations can assist the task of anomaly mitigation, such as the definition of proactive and reactive path instantiation, deployment of specific or generic flow rules in flow tables, and management of flow parameters such as timeout and data rate. Our framework must be able to instruct the network controller to change its behavior regarding certain events and flows as they are deemed anomalous.

We anticipate that our management framework must be modular and support customization, *i.e.*, it should be possible to update components with a more sophisticated algorithm or strategy whenever needed. For example, a *network driver* (see Section III-C) may need to be customized to collect information from different types of individual network controllers or from a large set of distributed controllers. Using the management plane, these decisions can be taken by administrators to achieve more appropriate configurations.

B. Lightweight and Heavyweight Processing

ATLANTIC comprises two operational phases: a *lightweight processing phase*, responsible for traffic monitoring and anomaly detection; and a more *heavyweight processing phase*, consisting of anomaly classification and mitigation. Next, we explain these phases in details and how they are combined to support robust anomaly management. Figure 1 summarizes the interplay between the lightweight and heavyweight processing phases.

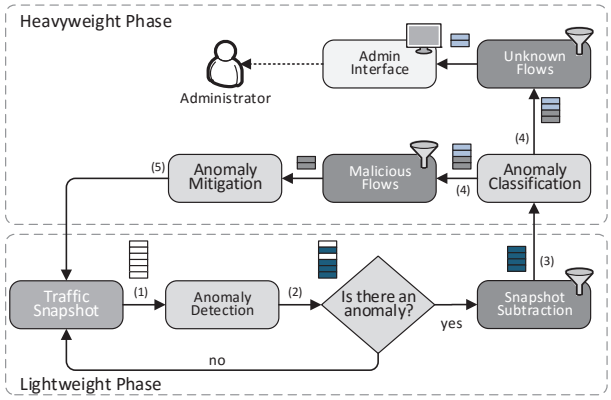


Fig. 1. Framework management process

1) Lightweight Processing Phase : Several techniques can be used to extract network traffic profiles, for example, by performing packet sampling using sFlow [17]. A limitation of this approach is the high memory consumption to obtain fine-grained traffic information and packet inspection. To rapidly perform lightweight anomaly detection, our framework benefits from the characteristics of SDN and uses the control plane to obtain a snapshot of existing traffic flows, including information about traffic counters and matched packet headers. Based on the traffic snapshot collected (arrow 1 in Figure 1), lightweight anomaly detection mechanisms are employed to detect deviations from the "normal" traffic pattern.

We apply entropy analysis to detect variations in the distribution of certain flow features observed along consecutive traffic snapshots. For example, consider two consecutive snapshots t_1 and t_2. If the entropy of the flow features in t_1 is approximately equal to the entropy of the same features observed in t_2, then it is safe to assume that no significant traffic changes have occurred between the two consecutive snapshots. However, if there is a large difference in the entropy calculated for a given flow feature between two snapshots, this might indicate an anomaly (arrow 2). If by subtracting the flows in t_2 from the flows in t_1 we obtain a non-empty result, this indicates which flows are responsible for the entropy change. Flows that are responsible for the entropy change in this stage can only be considered suspicious, and are thus selected for further categorization using a proper classification scheme (arrow 3). It is important to emphasize that our framework is designed so that any snapshot-based anomaly detection scheme can be employed. We chose to use entropy analysis for the *Lightweight Processing Phase* because it can be executed very often and permits fast detection of disturbances in the network [17].

2) Heavyweight Processing Phase: Our framework comprises the occasional need to execute complementary heavy processing classification mechanisms to categorize traffic flows. In this paper, we consider traffic classification mechanisms based on machine learning, which indeed take a considerable amount of resources to execute but can produce very accurate results in terms of traffic classification [12]. Our framework allows both supervised and unsupervised machine learning mechanisms. With supervised mechanisms, a model is generated to internally organize data obtained from previous malicious activity to automatically categorize traffic flows as either malicious or benign. Unsupervised mechanisms, on the other hand, are interesting to be used to analyze and organize information about traffic features, even if they cannot identify whether there is a threat or not. As a result, the framework allows flows to be categorized into either malicious, benign, or unknown, according to their traffic profiles. Malicious flows are sent for mitigation, whereas unknown flows need to be manually analyzed by a human administrator (arrow 4).

For every flow that is signalized as malicious, an action must be taken so to avoid network disruption or performance degradation. For example, commands can be sent back to the network control plane to instruct the devices closer to the source of the malicious traffic to drop packets of flows deemed malicious. After mitigation actions are performed, the framework returns to its initial traffic monitoring and snapshot collection step (arrow 5). For flows signalized as unknown, the administrator can use the information obtained during classification, for example, to create new models for the supervised mechanisms to identify the new traffic pattern in a future round

of anomaly detection. Note that this heavyweight phase is expected to be executed less frequently than the lightweight one. In addition, heavy classification mechanisms only need to deal with a subset of the full traffic snapshot, because of the subtraction performed in the lightweight phase. The more the administrator interacts with the framework inserting new information about traffic patterns to be automatically identified, the more efficient the detection will be.

C. Anomaly Traffic Classification

The basic components of our anomaly traffic classification framework are depicted in Figure 2. The *Statistical Layer* is responsible for collecting traffic flow statistics and comprises the following components: *Statistics Manager*, *Features Selector*, and *Network Driver*. The information generated by the *Statistical Layer* is delivered to the *Classification Layer*, which comprises the following components: *Anomaly Monitor*, *Flow Classifier*, and *Flow Manager*. Next, we describe these components in details.

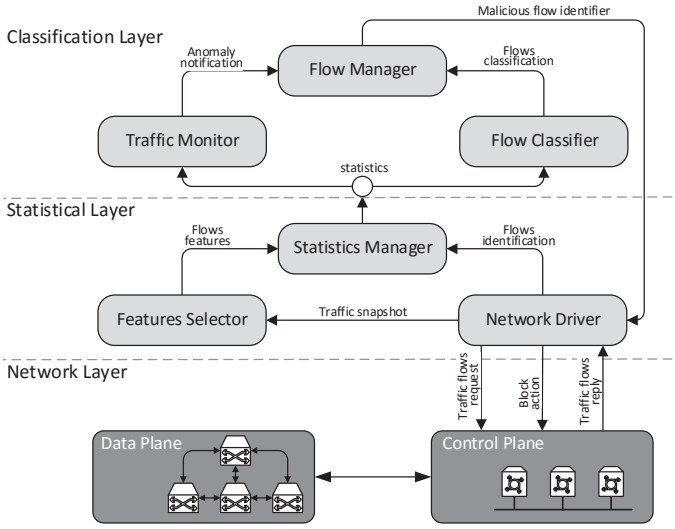


Fig. 2. Overview of the anomaly classification framework

1) Network Driver: The Network Driver operates by sending a request to the network controller every p seconds in order to query the status of all flows in the data plane. The parameter p can be adjusted accordingly, to avoid degrading network performance. After receiving flow information from the network controller, the *Network Driver* parses and organizes relevant data, such as the flow identifier, packet headers, and flow counters. The result of one request produces a traffic snapshot, *i.e.*, a data structure summarizing all flows currently existing in the network. Note that we construct a *flow-id* using the following 5-tuple, which is also used as our flow definition:

$$\langle srcip, dstip, srcport, dstport, protocol \rangle$$

2) Features Selector: After receiving the last saved traffic snapshot produced by the *Network Driver*, the *Features Selector* extracts basic flow features that describe the profile of network flows. The basic set of features is defined by the OpenFlow specification [4] and is represented by:

$$\langle packet\_count, byte\_count, duration \rangle$$

This component initiates the selection of relevant features. Frequently, the best choice for a flow profile model is a set of features that can accurately be used to discriminate a flow class with minimum computational cost. The issue of finding the optimal set of features to describe a flow can be addressed by manual inspection, or with the assistance of techniques such as PCA (*Principal Component Analysis*) and genetic algorithms [12]. This component can benefit from extensions to the controller API, which may enable additional information about flows to be exported.

3) Statistics Manager: The Statistics Manager summarizes all data collected by the *Network Driver* and by the *Features Selector* in order to derive statistical information that is used by the classification algorithms. In general, raw features have limited contribution to classification schemes without post-processing and statistical interpretation. After the features for one traffic snapshot are obtained, the *Statistics Manager* calculates *(i)* mean, *(ii)* standard deviation, *(iii)* coefficient of variance, and *(iv)* minimum and *(v)* maximum values as statistical discriminators to describe the data collected for each flow.

4) Anomaly Monitor: The main objective of this component is to monitor traffic flows using their statistics and basic flow features to detect potential anomalies. To exemplify how this component can behave, we use entropy analysis to detect changes in traffic features. In particular, this is calculated according to *Shannon's entropy* definition. Considering that a traffic snapshot is an alphabet, then the mean information $H(X)$ for some subset of features can be calculated using the available flows. We chose to calculate the entropy based on IP address and transport port features because they have been demonstrated to be accurate for the detection of DDoS attacks and worm propagations [11]. Every time that a new entropy E is calculated for a given snapshot, it can be classified as anomalous in the following way: considering that M represents the mean entropy observed in the network and S the standard deviation associated, then E will be an anomalous entropy if it is not within the interval $[M - S, M + S]$.

5) Flow Classifier: Different algorithms can be used for flows classification. When this component receives a set of feature statistics, a range of classification schemes (*e.g.*, machine learning algorithms) can run independently over the flow features. Note that this component is responsible for defining the class of each specific flow, by deciding which class is the most frequent when considering all algorithm outputs. Currently, we apply K-means [8] for clustering and Support Vector Machine (SVM) [22] for classification. We consider these algorithms suitable to initially exemplify our classifier because one can complement the results offered by the other, and the union of their outputs can be easily performed. Still, the *Flow Classifier* is customizable and can be extended with

additional algorithms[2].

6) Flow Manager: Events from the *Anomaly Monitor* and *Flow Classifier* are sent to the *Flow Manager* to indicate if an anomaly has been identified for a specific *flow-id*. Thus, the *Flow Manager* is responsible for deciding the mitigation actions to be taken when a malicious flow is identified. In this paper, we consider that a *'Malicious flow identifier'* message is sent to the *Network Driver*, indicating that the flow identified as malicious should be blocked. The *Network Driver* component further uses the *'Block action'* message to install firewall rules in the data plane and then modify how this plane handles the malicious flow. An example of action that could be taken by the *Network Driver*, as an alternative to dropping packets associated with malicious flows, is to forward such packets to another component (*e.g.*, a deep-packet inspector).

IV. Framework Evaluation

In this section, we present an experimental evaluation of ATLANTIC. The experiments were divided in two scenarios featuring different attacks: a worm propagation and a large-scale DDoS attack. We analyzed the amount of memory used by ATLANTIC and the processing time needed to provide anomaly detection, classification and mitigation. Our evaluation includes the study of *(i)* the performance of the lightweight phase; *(ii)* the performance of the heavyweight phase; and *(iii)* the classification accuracy of the overall framework.

A. Testbed and Simulation Profile

ATLANTIC was implemented on top of the Floodlight controller[3]. Floodlight provides a JSON-based REST API that is suitable for implementing the communication between the network controller and ATLANTIC. We run the experiments using the Mininet emulator. We used a topology of a campus network. It is a partially mesh topology consisting of 100 hosts and 11 switches. We chose this scenario because of recent malicious attempts to attack similar environments [23].

Table I describes the background traffic used in our experiments. When the simulation is set up, two services are configured: a video streaming server (whose service requests follow a lognormal distribution) and a Web server (whose service requests follow an exponential distribution) [24]. These services enable hosts to receive streaming over HTTP or send requests for Web pages, thus generating traffic related to file transfer. These traffic profiles have been validated in [24]. We generate these traffic profiles using VLC[4] for video streaming and SimpleHttpServer[5] to emulate an HTTP server.

In order to simulate the user behavior, we set up a scenario where users watch a video for a certain amount of time, pause it, and then access a few Web pages. For each group of 6 hosts requesting HTTP traffic, there is 1 host requesting video streaming traffic. Network anomalies related to malicious activities are generated with the *scapy* tool,[6] which enables the generation of realistic malicious attacks, such as port scanning and DDoS. The importance of these attacks has increased thanks to recent uses of DDoS to compromise campus communications [23]. Next, we explain the behavior of our simulated attacks.

TABLE I
Background traffic profile used in the experiments

Parameter	Value
Number of hosts	100
Number of switches	11
Number of servers	2 (HTTP and Streaming)
Number of attack flows	3500
Traffic profile	Video: 75 %, Web: 25 %
Host behavior Web Server	Exponential Distribution ($\lambda = 0.033$,mean = 30 s)
Host behavior Video Stream	Lognormal Distribution ($\mu = 11.75$,mean = 324 KB)

- **Port scanning -** A malicious host can use port scanning to discover a set of open ports in a remote host. Open ports can be used to exploit vulnerabilities in a target system or be used in worm propagations [25]. We simulate a malicious user that chooses a random host to start its attack to a server in the network. The attack consists of sending several TCP connection packets to ports ranging from 0 to 65536. When an open port is found, a notification is generated and this port can be used for worm propagations. Typically, *port scanning* generates packets in the network with a fixed IP address, but with varying transport protocol port.
- **DDoS attack -** We also defined a *DDoS attack* [11] scenario. A DDoS attack in SDN can be used to overflow with a large amount of spurious flows a specific switch's flow table or to overload the network controller by producing several `packet_in` messages. Frequently, these attacks result in a range of source IP addresses accessing a single target IP. In this case-study, we create a SYN flood attack, *i.e.*, a malicious machine chooses a server (HTTP or streaming) to send multiple TCP SYN packets to service ports offered by this machine (8080 for the VLC streaming and 8000 for the HTTP server). Given that there are services running in those ports, the server will process a request, allocate resources to handle it, and send an ACK to the requesting machine. Further, the attacker simulated in our experiments uses source IP spoofing, *i.e.*, he or she sends TCP SYN packets with the source IP address of another machine, thus causing the erroneous receipt of an unsolicited ACK and leaving the server with an open TCP connection.

B. Lightweight Anomaly Detection Evaluation

In this section we demonstrate the operation of the lightweight phase when it is instantiated with an entropy-based anomaly detection scheme to monitor the network in

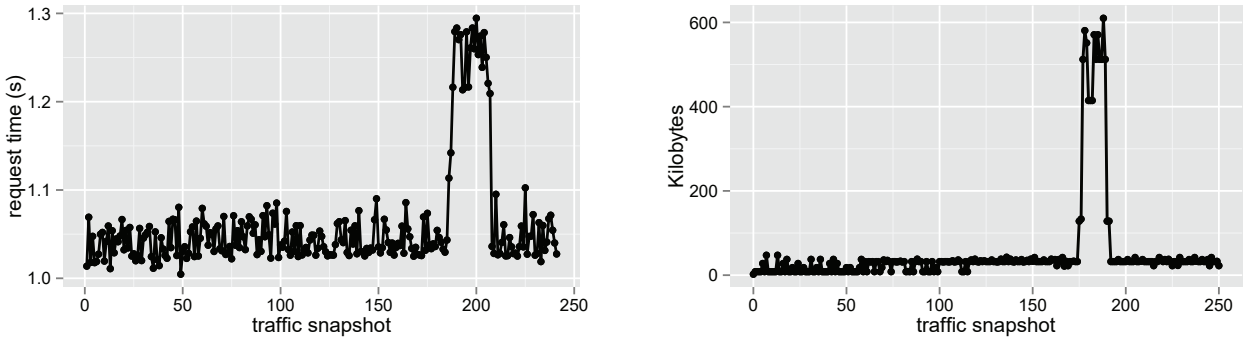
[2]In our implementation, the training phase for SVM is performed using simulated attack traces (∼100 malicious flows) in order to provide a range of samples to the classification schemes. Also, we use the R tool (https://www.r-project.org) to provide a reliable implementation of such machine learning algorithms.

[3]http://www.projectfloodlight.org/

[4]http://www.videolan.org/vlc/

[5]https://docs.python.org/2/library/simplehttpserver.html

[6]http://www.secdev.org/projects/scapy/



(a) Traffic snapshot requesting time

(b) Traffic snapshot memory used

Fig. 3. Resources usage to request and store a traffic snapshot

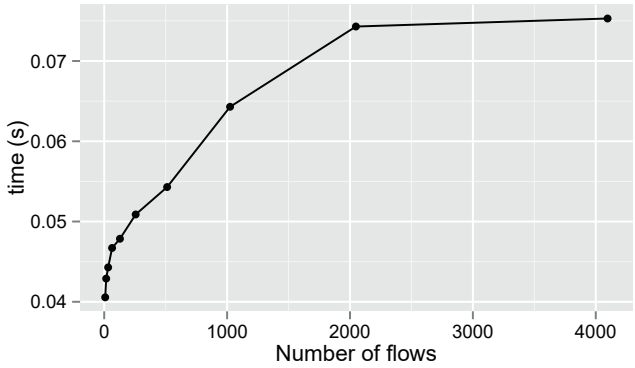


Fig. 4. Processing time of entropy calculation

the presence of malicious anomalies. Additionally, we analyze the performance of this phase regarding memory usage and processing time.

Initially, the *Network Driver* communicates with the controller to request traffic information. There are two possible bottlenecks in this approach: the traffic snapshot transmission time and the amount of memory needed to store this information. To understand these issues, we monitored the number of flows generated in the network while users were accessing HTTP pages and video streaming during a few minutes. Next, we started a DDoS attack and monitored the amount of new traffic flows. The transmission time required to export this information to our framework and the amount of memory needed were observed. The polling interval was set to 5s in order to obtain a fine-grained view of the network traffic. According to Figure 3(a), the transmission time related to traffic snapshot when an attack is not happening remains under 1.07s. Around the 180^{th} snapshot, the DDoS attack starts and increases this transmission time to 1.28s in the worst case. Furthermore, according to Figure 3(b), the size of network snapshots increase from 32 to 600 kilobytes with approximately 4,400 flows, including malicious and benign.

We argue that our simulations can realistically reflect the size of a large-scale campus scenario. In particular, it has been shown in [24] a similar performance evaluation that comprises around 800 traffic flows on average. We also analyzed individual flow rules in ATLANTIC and verified that a single flow rule uses around 136 bytes and takes less than 0.0008s to be transmitted. Thus, in order to simulate the campus scenario presented in [24], ATLANTIC would use 108,800 bytes (106.25 Kb) to store a traffic snapshot and 0.64s to transmit this information.

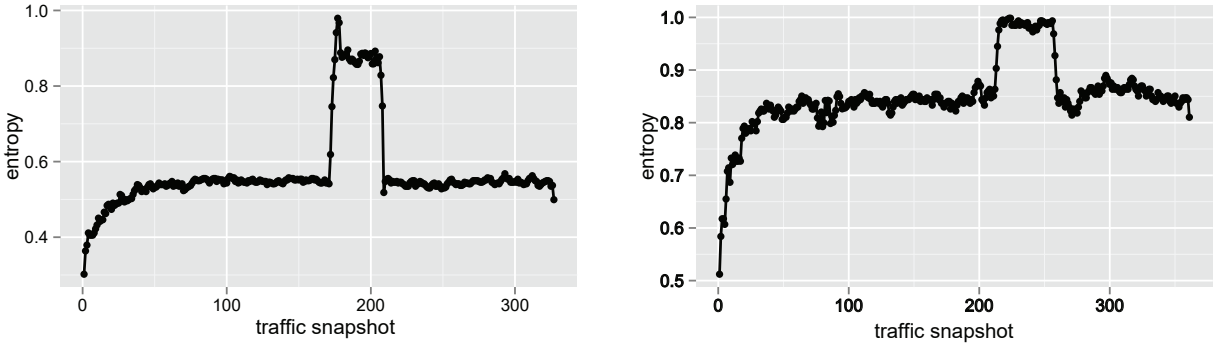
Entropy mean values associated with traffic features[7] can be used to detect network anomalies. The performance of the *Flow Monitor* itself is related to entropy calculation time, which grows linearly accordingly to the number of flows in a specific traffic snapshot. Figure 4 indicates that the entropy calculation time for 4,000 flows is 0.075s. We conclude that this type of traffic monitoring is suitable for ATLANTIC mainly because it is fast and accurate for detecting traffic deviations when we analyze, for example, the source IP entropy of a traffic snapshot. Figure 5(a) illustrates the variation in entropy of the destination port when a port scanning attack occurs. Around the 180^{th} snapshot, the average entropy that was around 0.55 rises to almost 1, indicating an anomaly. It is also possible to observe in Figure 5(b) the changes in the entropy of the destination IP address caused by the DDoS attack. Around the 280^{th} simulation snapshot, the DDoS attack stops, thus causing the entropy to revert back to normal (between 0.8 and 0.9).

When changes in the entropy of a specific feature are detected, the *Flow Classifier* component needs to determine the nature of existing flows in the network. The performance of this component is discussed in the next sub-section.

C. Heavyweight Anomaly Classification Evaluation

Each time a sudden change in the measured entropy of a specific feature is detected, a notification is generated and the ATLANTIC framework enters its heavyweight phase. The first action taken by the *Flow Classifier* is the classification of all remaining flows after the subtraction of the current traffic

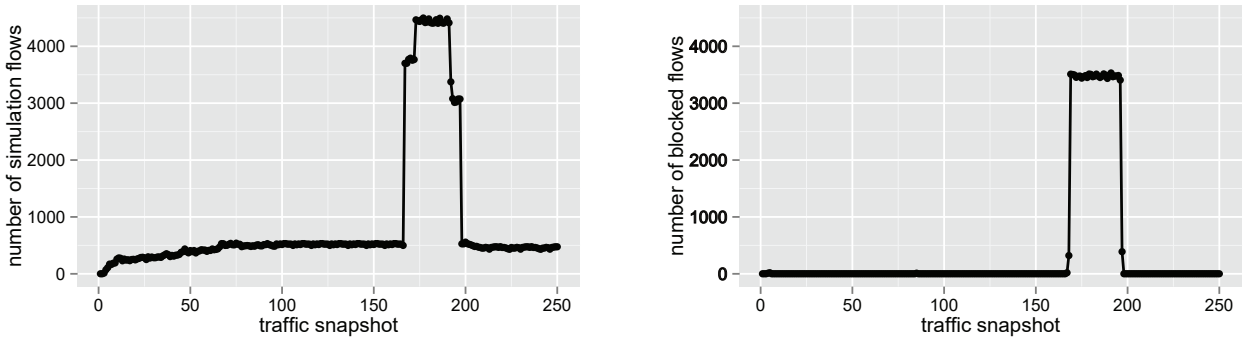
[7]We performed entropy analysis on all the features of a flow, but due to space constraints we only show results for destination ports and destination IP addresses in the first and second case-study, respectively.



(a) Destination port entropy in the port scanning scenario.

(b) Destination IP entropy in the DDoS attack scenario.

Fig. 5. Entropy observed including benign and malicious flow



(a) Overall flows

(b) Blocked flows

Fig. 6. Number of simulation flows

snapshot from the last one. Due to space limitations, we only show the results for the DDoS scenario.

To classify flows into separate classes representing DDoS and normal traffic, we instantiate the *Flow Classifier* with two well-known algorithms: K-means and SVM. Using K-means, similar flows are clustered together such that each cluster represents a type of traffic profile observed in the network. After this, the SVM algorithm can determine the classification of flows in each cluster. Note that each traffic snapshot may contain HTTP and streaming flows with different profiles, such as short-lived HTTP flows and long term video streaming flows. Figure 6(a) and Figure 6(b) summarize the amount of active flows in the simulation, as well as the flows signalized as malicious and subsequently blocked. Based on the classification offered by K-means or SVM, it is possible to calculate metrics such as *precision (PPV)* and *accuracy (ACC)*, which allow assessing the quality of the classification achieved by these algorithms[8]. In particular, it is possible to apply K-means using different values of k in order to find the optimal configuration and, jointly with SVM, find which combination achieves better classification metrics. Figure 7 illustrates our results. It can be observed that SVM presents accuracy of 88.7% and precision of 82.3%. The simulations were executed 35 times until the error rate was less than 0.01. The results obtained are very similar to the values expected from traditional networks, as presented in [26].

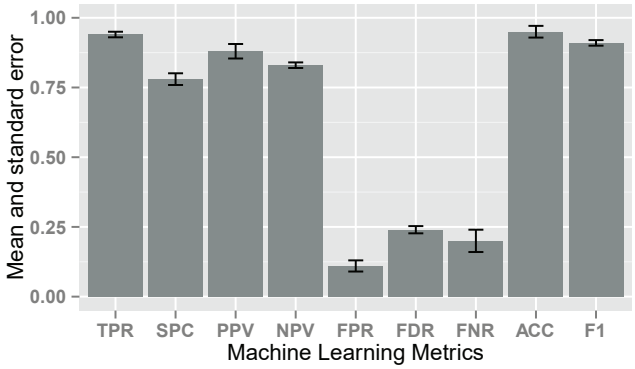


Fig. 7. Machine Learning metrics for SVM

After obtaining the classification, the *Flow Manager* is notified and then it is able to block malicious flows and restore the

[8]We computed the standard metrics commonly used in the evaluation of machine learning algorithms – TPR: sensitivity or true positive rate; SPC: specificity or true negative rate; NPV: negative predictive value; FPR: false positive rate; FDR: false discovery rate; F1: f1 score (harmonic mean of precision and sensitivity)

entropy back to normal. To block these malicious occurrences in the network, the *Flow Manager* can instruct the *Network Driver* to use a firewall application or to use the OpenFlow drop action installed on the data plane. This solution can block external malicious attacks from other networks.

The heavyweight phase lasts around 3 seconds in our simulations (Figure 8). When we compare this with the processing time of the lightweight phase (which takes around 0.07 seconds), we can more clearly appreciate the benefits of using more frequently the lightweight phase instead of always using the heavyweight phase to classify every traffic snapshot.

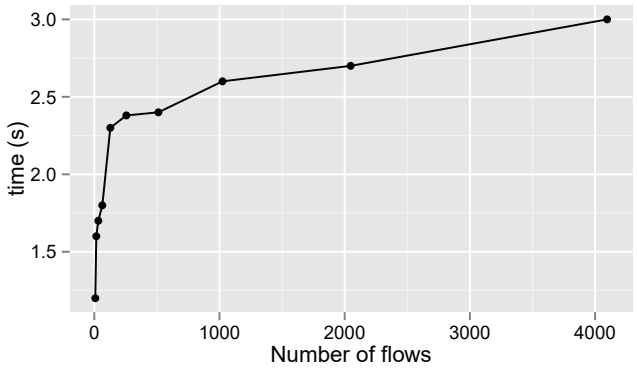


Fig. 8. Processing time of Heavyweight Phase

D. Discussion

We advocate that ATLANTIC meets the management requirements listed in Section IV.A. The lightweight step performs traffic monitoring using a global network view, demonstrating that detailed traffic information can be easily obtained. This contributes toward the comprehensive network view requirement needed for accurate anomaly detection.

The design of ATLANTIC allows the network administrator to monitor and modify the operation of its components. This characteristic contributes to a more tailored anomaly classification, which can rely on human intervention when the automated components are not able to protect the network. Additionally, using the network controller, ATLANTIC can orchestrate all flows in the network, sampling or eventually blocking a particular flow whenever needed.

V. Conclusions and Future Work

This paper proposed ATLANTIC, a framework for anomaly detection, classification and mitigation in SDN-based networks. Our framework comprises a lightweight phase responsible for monitoring traffic flows and a heavyweight phase responsible for anomaly classification and mitigation. As a result, traffic anomalies can be categorized and the information collected can be used to handle each traffic profile in a specific manner, such as blocking malicious flows.

In our experiments, the lightweight monitoring scheme enabled ATLANTIC to detect malicious activities without overloading the network, taking about 0.075s to collect and analyze traffic information consisting of 4400 flows in a topology with 100 switches. The heavyweight phase uses a machine learning algorithm (*i.e.*, SVM) which took less than 3s to classify traffic flows, demonstrating that ATLANTIC performs well even in the presence of DDoS attacks. Moreover, ATLANTIC executes the lightweight phase more frequently than the heavyweight phase, thus minimizing the overhead of the overall anomaly detection scheme. Most importantly, our results show how a sophisticated anomaly detection framework can be built over SDN.

As part of our future work, we aim to evaluate the use of different algorithms for traffic classification and entropy analysis to enforce network protection. We intend to investigate new mitigation strategies, such as the use of rate limiters, and new classification schemes, such as the combination of several network classifiers using meta-learning techniques (*e.g.,* stacking, and bayesian networks).

References

[1] J. P. G. Sterbenz, D. Hutchison, E. K. Çetinkaya, A. Jabbar, J. P. Rohrer, M. Schöller, and P. Smith, "Resilience and Survivability in Communication Networks: Strategies, Principles, and Survey of Disciplines," *Comput. Netw.*, vol. 54, no. 8, pp. 1245–1265, Jun. 2010.

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

[3] H. Kim and N. Feamster, "Improving Network Management with Software Defined Networking," *IEEE Communications Magazine*, vol. 51, no. 2, pp. 114–119, February 2013.

[4] Open Networking Foundation, "OpenFlow Switch Specification - Version 1.0.0 (Wire Protocol 0x01)," December 31 2009, available at: <https://www.opennetworking.org/sdn-resources/onf-specifications/openflow>. Accessed: August 2014.

[5] G. Nychis, V. Sekar, D. G. Andersen, H. Kim, and H. Zhang, "An Empirical Evaluation of Entropy-based Traffic Anomaly Detection," in *8th ACM SIGCOMM Conference on Internet Measurement (IMC)*, New York, USA, 2008, pp. 151–156.

[6] R. Yuan, Z. Li, X. Guan, and L. Xu, "An SVM-based Machine Learning Method for Accurate Internet Traffic Classification," *Information Systems Frontiers*, vol. 12, no. 2, pp. 149–156, Apr. 2010.

[7] T. Ide, S. Papadimitriou, and M. Vlachos, "Computing Correlation Anomaly Scores Using Stochastic Nearest Neighbors," in *7th IEEE International Conference on Data Mining (ICDM)*, 2007, pp. 523–528.

[8] J. Erman, M. Arlitt, and A. Mahanti, "Traffic Classification Using Clustering Algorithms," in *SIGCOMM Workshop on Mining Network Data (MineNet)*, New York, NY, USA, 2006, pp. 281–286.

[9] H. Kim, K. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet Traffic Classification Demystified: Myths, Caveats, and the Best Practices," in *ACM CoNEXT*, Madrid, Spain, 2008, pp. 11:1–12.

[10] W. Wang and S. Gombault, "Efficient Detection of DDoS attacks with Important Attributes," in *3rd International Conference on Risks and Security of Internet and Systems (CRiSIS)*, Oct 2008, pp. 61–67.

[11] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS Flooding Attack Detection Using NOX/OpenFlow," in *IEEE 35th Conference on Local Computer Networks (LCN)*, Washington, DC, USA, 2010, pp. 408–415.

[12] T. Nguyen and G. Armitage, "A Survey of Techniques for Internet Traffic Classification using Machine Learning," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 56–76, 2008.

[13] A. W. Moore and D. Zuev, "Internet Traffic Classification Using Bayesian Analysis Techniques," in *ACM SIGMETRICS*, New York, NY, USA, 2005, pp. 50–60.

[14] N. Williams, S. Zander, and G. Armitage, "A Preliminary Performance Comparison of Five Machine Learning Algorithms for Practical IP Traffic Flow Classification," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 5, pp. 5–16, Oct. 2006.

[15] A. Lakhina, M. Crovella, and C. Diot, "Mining Anomalies Using Traffic Feature Distributions," in *ACM SIGCOMM*, New York, NY, USA, 2005, pp. 217–228.

[16] Y. Yu, M. Fry, A. Schaeffer-Filho, P. Smith, and D. Hutchison, "An Adaptive Approach to Network Resilience: Evolving Challenge Detection and Mitigation," in *8th International Workshop on the Design of Reliable Communication Networks (DRCN)*, 2011, pp. 172–179.

[17] 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.

[18] B. Agarwal and N. Mittal, "Hybrid Approach for Detection of Anomaly Network Traffic using Data Mining Techniques," *Procedia Technology*, vol. 6, pp. 996 – 1003, 2012, 2nd International Conference on Communication, Computing and Security (ICCCS).

[19] Y. Zhang, "An Adaptive Flow Counting Method for Anomaly Detection in SDN," in *9th ACM Conference on Emerging Networking Experiments and Technologies*, ser. CoNEXT '13. New York, USA: ACM, 2013, pp. 25–30.

[20] S. Shin, P. Porras, V. Yegneswaran, M. Fong, G. Gu, and M. Tyson, "FRESCO: Modular Composable Security Services for Software-Defined Networks," in Internet Society NDSS, 2013.

[21] S. A. Mehdi, J. Khalid, and S. A. Khayam, "Revisiting Traffic Anomaly Detection Using Software Defined Networking," in *14th International Conference on Recent Advances in Intrusion Detection (RAID)*, Berlin, 2011, pp. 161–180.

[22] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988.

[23] J. J. Santanna, R. van Rijswijk-Deij, A. Sperotto, R. Hofstede, M. Wierbosch, L. Zambenedetti Granville, and A. Pras, "Booters - An Analysis of DDoS-as-a-Service Attacks," in *14th IFIP/IEEE International Symposium on Integrated Network Management (IM)*, Ottawa, Canada, May 2015, pp. 243 – 251.

[24] P. H. Isolani, J. A. Wickboldt, C. B. Both, J. Rochol, and L. Z. Granville, "Interactive Monitoring, Visualization, and Configuration of OpenFlow-based SDN," in *14th IFIP/IEEE International Symposium on Integrated Network Management (IM)*, Ottawa, Canada, May 2015, pp. 207–215.

[25] A. Schaeffer-Filho, A. Mauthe, D. Hutchison, P. Smith, Y. Yu, and M. Fry, "PReSET: A Toolset for the Evaluation of Network Resilience Strategies," in *13th IFIP/IEEE International Symposium on Integrated Network Management (IM)*, Ghent, Belgium, May 2013, pp. 202–209.

[26] X. Li, F. Qi, D. Xu, and X. Qiu, "An Internet Traffic Classification Method Based on Semi-Supervised Support Vector Machine," in *IEEE International Conference on Communications (ICC)*, Kyoto, Japan, June 2011, pp. 1–5.