vSwitchGuard: Defending OpenFlow Switches against Saturation Attacks

Samer Y Khamaiseh
Department of Computer Science
Midwestern State University
Wichita Falls, TX, USA
Samer.khamaiseh@msutexas.edu

Edoardo Serra
Department of Computer Science
Boise State University
Boise, ID, USA
edoardoserra@boisestate.edu

Dianxiang Xu
Dept. of Computer Science Electrical Eng.
University of Missouri – Kansas City
Kansas City, MO 64110, USA
dxu@umkc.edu

Abstract— **While the decoupling of control and data planes in software-defined networking (SDN) facilitates orchestrating network traffic, it suffers from security threats. For example, saturation attacks can make SDN out of service by exhausting the controller' and switch's computational resources. The existing research has focused on defense against limited types of saturation attacks. In this paper, we propose vSwitchGuard, a framework for detection and countermeasure of known and unknown saturation attacks in SDN. vSwitchGuard aims to identify the victim switches targeted by known or unknown types of saturation attacks with machine learning classifiers and restore the victim switches to their safe state through deep packet inspection. We have evaluated three supervised classifiers and four semi-supervised classifiers for five types of saturation attacks (TCP-SYN, UDP, ICMP, IP-Spoofing, and TCP-SARFU) and their combinations. The results suggest that supervised and semi-supervised classifiers can be combined to deal with known and unknown attacks for better performance. We have also implemented the countermeasure and evaluated it with all combinations of the five types of attacks. The results demonstrate that vSwitchGuard can effectively defend against the attacks without significant performance overhead.**

Keywords- software-defined networking; OpenFlow; saturation attack; machine learning; DoS attacks

I. INTRODUCTION

Software-Defined Networking (SDN) is a programmable network architecture that separates the control plane from the data plane. The components in the data plane, e.g., OpenFlow switches, are responsible for forwarding network traffic. The controller in the control plane is a centralized unit that translates the application requirements down to the data plane by using a southbound interface. As the first proposed communication protocol between the data and control planes, OpenFlow has been adopted as the standard southbound APIs by Open Network Foundation (ONC) [1].

The reactive packet processing approach of OpenFlow makes an SDN network more adaptable to requirement changes of applications. An OpenFlow switch processes each incoming packet by matching it with flow rules in the flow tables. When there is no match (i.e., table-miss), the switch encapsulates the packet inside a Packet-In message and sends it to the controller. The controller then decides a proper action and, if necessary, installs a new flow-rule in the OpenFlow switch. Such a reactive approach to packet processing, however, can be exploited by saturation attacks that send maliciously spoofed packets. The table-misses will trigger a large number of Packet-In messages sent to the controller by the switch. This may exhaust the computational recourses of the control plane and congest the OpenFlow channel between the data and control planes. The large number of fake flow entries created by the controller may also exhaust the flow-table buffer of the switch. As a result, the entire network may be paralyzed.

Several approaches have been proposed to deal with saturation attacks. AVANT-GUARD [2], SLICOTS [3], and LineSwitch [4] focused on SYN flooding, a classical method of denial of service attack in computer networks. FloodGuard [2] and FloodDefender [6] relied on the rate of Packet-In messages to reason about attack occurrence. This can be inaccurate because the resultant rate of a high volume of normal traffic may also exceed a predefined threshold [7]. FDAM [8] used the Support Vector Machine (SVM) to detect attacks and a whitelist of IP addresses to prevent the attacks. SA-Detector [7] exploited self-similarity degree of OpenFlow traffic to detect several types of saturation attacks. It is based on the observation that OpenFlow traffic between the controller and switches has a relatively higher degree of self-similarity when a saturation attack occurs. Our recent work [9] has demonstrated that supervised classifiers are useful for detecting the occurrences of these types of attacks. However, the existing machine-learning based approaches have two main limitations. First, they deal with pre-defined (known) types of attacks using supervised classifiers. It is unclear if they are capable of detecting new attacks. Detection of unknown saturation attacks in SDN is important because research has shown that there are various attack methods. Second, they aim to detect whether or not a saturation attack is happening in an SDN environment, but do not determine which switches are being attacked. In other words, they cannot recover the victim switches from the actual occurrence of a saturation attack.

To address these issues, this paper presents vSwitchGuard, an effective framework for detection and countermeasure of both known and unknown types of saturation attacks in SDN. The contributions are as follows:

- In addition to the OpenFlow message types [9], this paper proposes the statistical properties of

978-1-7281-7303-0/20/$31.00 ©2020 IEEE
DOI 10.1109/COMPSAC48688.2020.0-157

OpenFlow message payloads (entropy and table-miss packet rate) as features for machine learning-based attack detection and show that the combination is the most effective.

- We evaluate and compare three supervised classifiers (SVM, K-Nearest Neighbor, and Naïve-Bayes) and four semi-supervised classifiers (Basic Autoencoder, Variational Autoencoder, One-Class SVM, and Isolation Forest) for locating the victim switches targeted by known and unknown attacks. Supervised and semi-supervised classifiers can be combined improve detection performance. Among the four semi-supervised classifiers, Variational Autoencoder is the most effective in dealing with unknown attacks.
- We present an effective countermeasure that restores the victim switches to a safe state. Based on deep packet inspection, the countermeasure can accurately block the malicious incoming traffic and clean up the flow-rules installed in the victim switches. Unlike some of the existing work, vSwitchGuard does not need to add new hardware or modify the SDN architecture.
- We present an online testing of the countermeasure with all combinations of the five attack types. The results show that vSwitchGuard can effectively defend the OpenFlow switches without significant performance overhead in terms of CPU utilization, OpenFlow channel bandwidth, and flow table utilization.

II. RELATED WORKS

Identifying targeted OpenFlow switches. Chi et al. [10] proposed a preliminary method that samples flow-rules from randomly selected OpenFlow switches. It generates artificial packets to see if the OpenFlow switch executes the corresponding flow-rules correctly. This approach may produce a high rate of false-positives since the flow-rules of the OpenFlow switches are changing over time.

Zhou et al. [11] proposed SDN-RDCD, a real-time approach to detect the targeted SDN devices when the controller and OpenFlow switches are trustless. It uses a backup controller as an audit controller that is responsible for recording the network update events such as deleting, adding, or updating flow rules from the original controller and its connected OpenFlow switches. The audit controller allocates a unique audit ID for each update request event and records it in an audit record. This audit ID is used to keep track of each event, as well as the execution results on the original controller and corresponding OpenFlow switches. The audit ID is also used by the audit controller to re-execute the update event and record the execution results. Then, SDN-RDCD analyzes the recorded audit log to extract any inconsistency of the handling of information by the controller and OpenFlow switches. This approach may require a long time to process the audit records in order to find the unmatched event handling information. Thus, the saturation attacks may have compromised the entire network before the targeted OpenFlow switches are detected. This approach cannot detect most of the switches that are targeted by saturation attacks, since the behavior of these targeted OpenFlow switches is very similar to the normal ones.

Different from the above work, vSwitchGuard provides an effective method that can detect the targeted OpenFlow switches by known and unknown attacks. It does not require any modification to the SDN architecture.

Detecting saturation attacks. Various approaches have been proposed to detect saturation attacks in SDNs [12]. Ashraf et al. [13] discussed the possibility of adopting machine learning approaches to detect DoS attacks in SDN. This work does not deal with victim switches. Quamar et al. [14] adapted the Stack Autoencoder (SAE) deep learning technique for detecting multi-vector DDoS. The proposed defense system consists of three components: Traffic Collector and Flow Installer, Feature Extractor, and Traffic Classifier. This work relies on processing every incoming packet for attack detection and flow computation, which requires extensive computational resources. The dataset was used for training and testing was collected from a traditional wireless network, so it does not involve OpenFlow traffic. Braga at al. [15] adopted the self-organized map (SOM) for detecting flooding attacks against SDNs. The detection system collects all the flow entries of the connected OpenFlow switches, extracts the SOM classifier features from the collected flow-entries to detect the DDoS attacks. However, this work requires extensive processing time to extract the features because it needs to process all the flow entries of connected OpenFlow switches. This delay may give the attacker enough time to flood the network. Abubakar and Pranggono [16] developed a flow-based anomaly detection system by using a neural network. The NSL-KDD dataset was from a traditional network, not SDN network. Tang et al. [17] used the Deep Neural Network (DNN) to develop an anomaly DoS detection system. The accuracy is relatively low – just 88.04%. It was also based on the NSL-KDD dataset from a traditional network.

Santos et al. [18] introduced the ATLANTIC framework to detect DDoS attacks against SDNs. It consists of two phases: (1) a lightweight processing phase that can be executed periodically to detect the deviations of the SDN network traffic flows by using entropy analysis in order to identify the suspicious traffic flows, and (2) a heavyweight processing phase that uses a K-means unsupervised algorithm to cluster the similar traffic flows and then adopts an SVM classifier to classify the malicious flows from the normal ones. The detection performance is relatively low: 88.7% accuracy and 82.3% precision. The approach also caused performance overhead on the SDN environment because it required a long prediction time to use SVM and K-means together.

Ye et al. [19] proposed a detection system for UDP, SYN, and ICMP flooding attacks by using an SVM classifier. The system includes: (1) a flow state collection module that collects the status of the OpenFlow switches flow-tables by using controller-to-switch messages, (2) Characteristic Values Extraction module that is responsible

for extracting the classifier features (it extracts 6 features from the collected flow tables' status messages), and (3) classifier judgment module, which utilizes an SVM classifier to detect the attacks. However, SVM is deficient in detecting unknown saturation attacks.

Li et al., [7] proposed a lightweight method called SA-Detector for detecting a family of saturation attacks. It is based on the study of self-similarity of OpenFlow traffic. It uses the Hurst exponent for normal and abnormal OpenFlow traffic to detect the saturation attacks. Our prior work [9] have studies several supervised classifiers to detect all the attack types in [7]. The above work, however, does not identify the targeted switches.

Countermeasure of saturation attacks. Several methods have been proposed to defend SDN against saturation attacks. Hu et al. [7] Introduced the FDAM system for detecting UPD, ICMP, and SYN flooding attacks. It consists of two modules: (1) an attack detection module that is responsible for detecting DoS attacks by using an SVM classifier and a sFlow approach to collect the network traffic and extract features, and (2) a DoS attacks mitigation module that mitigates flooding attacks by using traffic migration and white-list approaches. As shown in this paper, SVM does not have a good record in detection of the saturation attacks. Shang et al. [6] Proposed FloodDefender as an SDN application to protect the control plane and data plane against DoS attacks. It has four modules: attack detection to detect the DoS attacks, table-miss engineering to migrate the table-miss packets to the neighbors' switches, packet filtering to identify the attack traffic, and flow rule management to remove the useless flow-rules. The main limitations are: (1) All the table-miss packets are delivered to the controller, whether they are normal or not. (2) There is a relatively high flow-table utilization due to the installation of protecting and monitoring flow-rules. (3) The attack detection module using the Packet-In messages rate is ineffective because a high volume of normal traffic can produce a similar rate of Packet-In messages.

Wang et al. [5] proposed FloodGuard to protect the controller against DoS attacks. FloodGuard has two approaches to detect and mitigate DoS attacks: a proactive flow rule analyzer module to enforce the network functionalities when the DoS attack occurred and packet migration to protect the controller from being overloaded. Shin et al [2] proposed the AVANT-GUARD framework to mitigate the TCP-SYN flooding that is sent to the SDN controller. It accomplishes this task by extending the OpenFlow-Switches functions. The detection module monitors the ongoing TCP-SYN connections to the controller and detects the SYN flooding based on a predefined threshold, which cannot accurately differentiate between the normal and abnormal SYN packets. Both AVANT-GUARD and FloodGuard focus on protecting the control plane against DoS attacks and ignoring the data plane and the OpenFlow connection channel.

Different from the above work, vSwitchGuard can detect the targeted OpenFlow switches by known and unknown saturation attacks. We have studied different supervised and semi-supervised classifiers. vSwitchGuard also provides a countermeasure that can effectively mitigate a family of saturation attacks and recover the victim switches.

III. Background and Problem Statement

A. SDN and OpenFlow Protocol

The SDN architecture comprises the control and data planes that communicate through the southbound API (i.e., OpenFlow). The centralized controller translates applications' network requirements down to the data plane, which consists of network hardware components such as switches and routers. The southbound API, as an interface between the switches and the controller, enables the controller to manage the behavior of the entire network by installing flow-rule in the flow tables of switches.

There are 29 types of messages between the controller and OpenFlow switches. They fall into three groups: (1) controller-to-switch messages sent by the controller to update, add, delete group/flow entries or request the status of a switch, (2) asynchronous messages initiated by a switch to inform the controller of new packet arrival that does not match the flow entry rules or changes in the switch state, (3) symmetric messages initiated in either the direction of controller-to-switch or the direction of switch-to-controller. When receiving a new packet from an application, either normal or malicious, an OpenFlow switch tries to match it with the rules in the local flow table. If there is no match, a table-miss occurs, and the switch generates a Packet-In message. The message contains the header of the table-miss packet if the switch buffer is not full, otherwise the table-miss packet is encapsulated in the Packet-In message and sent to the controller. When receiving the Packet-In message, the controller decides how to process the table-miss packet and tries to install a new flow-rule in the flow table by sending Packet-Out and Packet-Mod messages.

B. Adversary Model

The aforementioned reactive approach to packet processing is subject to saturation attacks. An attacker may exploit table-misses to exhaust the computation resource (e.g., CPU and memory) of the controller and/or switches, and saturate the OpenFlow channel. The saturation effect is caused by the generation of a large number of Packet-In messages, which significantly consumes the controller's computational resources.

The saturation effect is caused by the generation of a large number of Packet-In messages, which significantly consumes the controller's computational resources. When the switch-to-controller communication is flooded by Packet-In messages, the controller will also send a large number of Packet-Out and Packet-Mod messages, which leads to the flooding of controller-to-switch communication. The flow tables in the affected switches are filled with fake flow rules. This may prevent the benign flow rules to be

installed and thus the system may be unable to process the legitimate new packets.

To implement saturation attacks in SDN environments, there are various ways to generate malicious network traffic to trigger a large volume of Packet-In messages. Examples are SYN flooding, UDP flooding, ICMP flooding, IP-Spoofing, and TCP-SARFU flooding [7]. Even worse, these methods can be combined to lunched very complex attacks. To defend against saturation attacks in SDN, it is important to deal with known and unknown types of malicious traffic.

C. Challenges

This paper aims at effective detection and countermeasure for defending OpenFlow switches from known and unknown saturation attacks. There are three main challenges: (1) how to detect unknown attacks, (2) how to determine which switches are the victims, and (3) how to restore the victim switches to a safe state.

In SDN, table-miss packets, and consequently Packet-In, Packet-Out and Packet-Mod messages, may originate from both benign and malicious applications. The controller and switches have inadequate knowledge to determine if table-miss packets are triggered by a benign or malicious application. From the qualitative perspective, the forwarding behaviors of OpenFlow switches do not follow a predefined norm. Nevertheless, OpenFlow messages may exhibit statistical properties when an SDN environment is under a saturation attack. For example, four of the 29 message types were used as the critical indicators of the existing saturation attacks [9]. It is thus feasible to apply the statistical information of these messages to detect saturation attacks with machine learning classifiers. However, the existing work has not addressed the first two challenges. It is unclear whether these classifiers and message types are effective for dealing with unknown attacks. Unlike traditional switches where forwarding rules are predefined, OpenFlow switches are programmed by the controller according to all relevant applications. As the flow rules of an OpenFlow switch dynamically change over time, the forwarding behaviors of an OpenFlow switch do not exhibit a single set of behavioral norms. It is non-trivial to determine if a switch is under attack and how to restore it to a safe state. In this paper, we address the first two challenges by applying and comparing supervised and semi-supervised classifiers to the OpenFlow messages between the controller and individual switches. In addition to the four critical message types, we also use the OpenFlow message payloads for attack detection. To determine if a classifier can recognize an unknown type of attack, the samples of a given attack type are excluded from the training data, but included in the testing data.

To restore victim switches, a countermeasure must be able to accurately identify and remove the fake flow rules installed during the attack and should not drop any legitimate table-miss packets from benign switches and hosts. As the controller may use various attributes of incoming OpenFlow packets for matching flow rules (port number, switch DPID, MAC address, and IPv4 addresses), the countermeasure requires deep packet analysis.

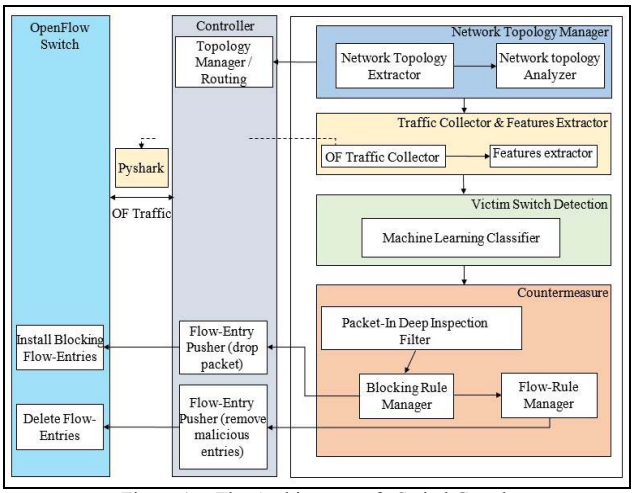

Figure 1. The Architecture of vSwitchGuard.

IV. SYSTEM DESIGN

vSwitchGuard is designed as an application on top of the controller, as depicted in Figure 1. It consists of four modules: network topology manager, OpenFlow traffic collector and feature extractor, victim switch detection, and countermeasure. The network topology manager extracts the SDN topology by using the northbound REST APIs of the controller and classifies the extracted network topology based on the connected OpenFlow switches. The traffic collector is a session-based process that collects the raw OpenFlow traffic by using the Pyshark library. The feature extractor extracts the features from the raw traffic data for machine-learning based detection of saturation attacks. The duration of each session is a pre-defined time-window for OpenFlow traffic analysis. In this paper, the time-window is set to one-minute suggested [9] in because the same dataset is used for the evaluations of both supervised and semi-supervised classifiers. The module for victim switch detection applies a supervised or semi-supervised classifier to the processed traffic feature to determine whether or each switch is under a saturation attack. The model of the classifier is obtained by training the classifier with an offline dataset. It is then used for online detection.

Once a switch is found to be under attack, the countermeasure module will invoke the deep inspection component to allocate the Packet-In messages from the collected traffic. It inspects the header of the table-miss packet inside the Packet-In messages to identify the zombie hosts. According to the results of inspection, the blocking flow-rule manager will install high priority blocking rules and the flow-table manager will identify and remove the fake flow rules according to the network topology.

V. DETECTION OF VICTIM SWICHES

This section first describes the statistical features of OpenFlow traffic used to detect attacks with classifiers, and then introduces the supervised and semi-supervised classifiers studies in this paper.

A. *OpenFlow Traffic Features*

The existing research has shown that, among the 29 types of OpenFlow messages, the following four are significantly affected by known saturation attacks [9]: (1) the number of Packet-In messages generated by a switch (2) the number of Packet-Out messages received by a switch, (3) the number of Packet-Mod messages received by a switch, and (4) the number of TCP-ACK messages received and generated by a switch. As these features are obtained from OpenFlow messages headers, we refer them to the features of OpenFlow message headers, which are useful for detecting known saturation attacks with supervised classifiers [9].

In this paper, we propose two new features, entropy and table-miss packet rate (TPR). They are referred to as OpenFlow message payload features because they utilize detailed information in OpenFlow messages. While investigating malicious and normal OpenFlow traffic, we observed that, when a switch is under saturation attack, the distribution of the source IPv4 addresses of the table-miss packets encapsulated inside the Packet-In message payload (i.e., data field) changed frequently. To measure such distribution changes, we calculate the Shannon Entropy value of source IPv4 addresses of the table-miss packets for each OpenFlow switch. Shannon Entropy is a measurement of the uncertainty of random variables in information theory. The entropy value of source IPv4 addresses of the table-miss packets is defined as:

$$E(srcIP) = -\sum_{i=1}^{k} (n_i/M)\log_k(n_i/M) \quad (1)$$

Where $srcIP = \{n_1, n_2, \ldots, n_k\}$ represents all the source IPv4 addresses of an OpenFlow switch table-miss packets encapsulated inside the Packet-In messages within the specified time-window, n_i is the occurrence number of the *ith* source IPv4 address IP_i , and K is the number of different sources IPv4 addresses. $M = \sum_{i=1}^{k} n_i$ represents the total occurrence number of all source IPv4 addresses of table-miss packets of an OpenFlow switch. A high (low) entropy value indicates a more decentralized (concentrated) probability distribution.

Figure 2 presents a sample of the total received source IPv4 addresses of normal and malicious packets. It also shows the number of normal and malicious table-miss packets source IPv4 addresses. In the malicious incoming packets within one minute of OpenFlow traffic, 574 of the 962 source IPv4 addresses triggered table-miss packets. In contrast, only 24 of the 684 source IPv4 addresses in the legitimate incoming packets were involved in table-miss packets. In this case, the entropy value of the malicious table-miss packets was higher than that of the legitimate table-miss packets.

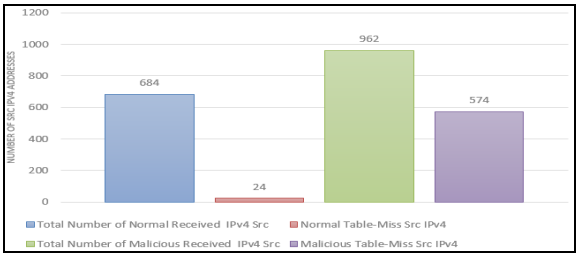

Figure 2. A Sample of Normal and Malicious Table-Miss Source IPv4 Addresses.

Table-Miss Packet Rate (TPR) is the proportion of table-miss packets (i.e., Packet-In messages) out of the total corresponding received packets of an OpenFlow switch within a specified time-window. It is defined as follows:

$$TPR = \sum S(PacketIn) / \sum S(Received\ Packets) \quad (2)$$

Where $\sum S(PacketIn)$ is the total number of generated Packet-In messages (the number of table-miss packets), and $\sum S(Received\ Packets)$ is the total number of received/incoming packets of the OpenFlow switch.

Figure 3 shows an example of normal and abnormal OpenFlow traffic of a switch, where the switch received 140,531 legitimate packets and generated 15,637 Packet-In messages, with a TPR value of 0.11. It indicates that 11% of the received packets caused a table-miss, and 89% of the received packets matched the flow table entries. In contrast, the same switch targeted by saturation attacks received 96,463 packets and generated 38,164 Packet-In messages, with a TPR value of 0.39 -- 39% of the received packets are table-miss packets.

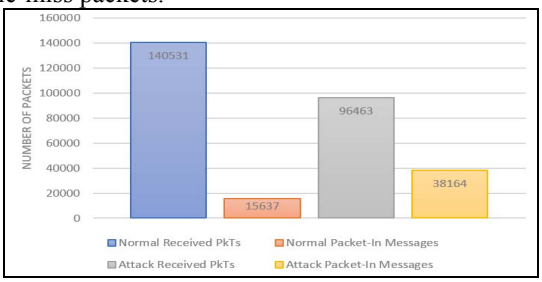

Figure 3. A Sample of Normal and Malicious Table-Miss Packets.

To summarize, this paper evaluates and compares three groups of OpenFlow traffic features: four message header features (Packet-In, Packet-Out, Packet-Mod, TCP-ACK messages), two message payload features (entropy and TPR), and the combination of them (i.e., all six features).

B. *Attack Detection with Classifiers*

In this paper, we studied and compared three supervised classifiers: K-Nearest Neighbors (K-NN), SVM, and Naïve Bayes (NB) classifiers and four semi-supervised classifiers: one-class SVM, isolation-forest, basic autoencoder, and variational autoencoder. Our prior work has used K-NN, SVM, and NB to detect whether or not an SDN system is under saturation attack [9], but it did not determine which switches is being attacked. In this paper, we not only use

the same supervised classifiers for detecting victim switches, but also evaluate their capability of dealing with unknown types of attacks. The supervised classifiers required that the training dataset include specimens of all saturation attack types. The patterns they learned from the training data tend to focus on the representation of malicious behaviors, assuming other samples in the dataset as benign behaviors. In other words, they may classify a new unknown attack as benign because the attack differs from the pattern learned from the training data. This has motivated the application of semi-supervised classers, which can be trained without the need to label the samples.

Our experiments have shown that, among the four semi-supervised classifiers, variational autoencoder has achieved the best performance. In the following, we provide a brief introduction. Autoencoder is an artificial neural network composed of two functions: the encoder and the decoder. The encoder function E is a neural network transforming the original features x in a new space $y=E(x)$. Usually, y is in a lower dimension than the original space. The decoder function D transforms the features Y from the new space to the original space $x'= D(x)$ This neural network is trained by minimizing the reconstruction error, i.e., $loss = || x- x' ||$. Once trained, the reconstruction error on a new sample can be used as a score to determine whether or not the sample represents a training example.

The smaller the reconstruction score, the higher the likelihood that the new sample is similar to the one used in the training. For two similar input vectors x and x', a basic autoencoder can generate very different encoding representations y, and y'. Thus, similar instances cannot be placed in the same encoded space since it may lead to poor detection performance. Variational autoencoders aim to address this issue, in a variational autoencoder, the basic autoencoder represents the encoding by a distribution of vectors rather than a single vector. The encoder network generates the mean vector $\mu = E_{\mu}(x)$ and the covariance matrix $\sum = E_{\sum}(x)$. They are used in a multivariate normal distribution $N(\mu, \Sigma)$ generating the encoding $\gamma \sim N(\mu, \Sigma)$. In addition to the standard reconstruction error, the variational autoencoder imposes that the distribution of the encoded vector is approximately similar to a normal distribution with mean zero and standard deviation (i.e., they use the encoding Kullback-Leibler divergence regularization term).

VI. COUNTERMEASURE

Countermeasure module works when the detection module has identified the victim switches of saturation attacks otherwise it remains idle. It is does not require any modification of the SDN design or any extra device. It can mitigate a family of saturation attacks by utilizing (1) the Packet-In deep inspection filter, which is responsible for allocating the attacking sources, targeted hosts, and reducing the false-positives of the victim switch detection module, (2) the blocking-rule manager component, which is responsible for blocking the malicious incoming traffic from the attacking source. (3) our countermeasure method can accurately identify the installed malicious flow-entries and remove them using the flow-table manager component.

1) Packet-In Deep Inspection Filter

Packet-In deep inspection filter can identify the zombie hosts, the targeted destination and reduce the false-positives of the victim switch detection module by inspecting the Packet-In messages of each victim switch. The working process of Packet-In message filter as follows:

- Allocate all the Packet-In messages of each switch that has been identified as a victim.
- Inspect the data filed for each Packet-In message, which contains the header of the table-miss packet or the whole table-miss packet. We extract the source and the destination IPv4 addresses, MAC addresses, and OpenFlow switch port numbers of the table-miss packets.
- Identify the zombie hosts and the target destinations by comparing the table-miss packets source and destination IPv4 addresses with the network topology. Essentially, the attacker keeps spoofing the content of the transmitted packets to reduce the possibility of matching any flow-rules in order to urge the targeted OpenFlow switch of generating Packet-In messages, specifically, the source IPv4 address of the table-miss packets. Since the malicious packets with the same IPv4 address will drastically downgrade the performance of saturation attacks. The zombie hosts can be identified by comparing the MAC address, port number, and source IPv4 address of the table-miss packet with the network topology. If source IPv4 address has zero matches with the network topology and the MAC address, and the port number matches with the network topology. The table-miss packet will be considered as a malicious one and the host with the corresponding source MAC address is a zombie host. Also, the destination of the table-miss packet is regarded as a targeted destination and the switch of the Packet-In messages will be regarded as a victim.

2) Blocking-Rule Manager

The blocking-rule manager triggered by the Packet-In deep inspection filter. It aims to mitigate the attack by blocking the malicious incoming traffic from the zombie hosts. After the zombies' hosts are identified, the blocking-rule manager obtains OpenFlow switch(s) of the zombies' hosts by comparing the zombie host MAC address and port number with the network topology. Then, it installs a high priority blocking-flow rule on the zombie host OpenFlow switch by using controller-to-switch messages.

The blocking rule consists of a switch DPID field which is equal to the zombie host OpenFlow switch DPID, a priority field that is used to assign the priority of the installed flow-rule. Since the processing of the flow-rules is

based on priority, the assigned value should be the highest. The MAC address field is set to the zombie host MAC address, the ingress port field is an OpenFlow switch port of the zombie host, and the action field is set to the "*Drop*" action to block the malicious incoming traffic. After installing the blocking-rule, the incoming malicious traffic from the ingress port will be matched against the blocking-rule. Thus, the incoming attack traffic will be blocked.

3) Flow-rule manager

One of the main destructive consequences of a saturation attack is preventing legitimate flow-entries from being installed. This happens because the attack consumes the victim switches memory by installing a huge amount of malicious flow-entries into their flow-tables. Therefore, allocating and removing the malicious flow-entries in the victim switch is an important step to recover the victim switch from the saturation attack.

When a table-miss occurred, the controller installs flow-entries into the OpenFlow switch flow-tables to match the new incoming traffic. The process of creating flow-entries does not exhibit a single behavior. The controller may use the port numbers, switch DPID, MAC addresses of the source and destination, IPv4 addresses of the source and destination, or their combinations to match the incoming traffic. Therefore, we have different forms of malicious flow-entries.

The flow-rule manager identifies the malicious flow-rules and removes them from the victim switch's flow tables. It consists of two phases:

- The flow-rule manager creates the saturation attack topology through obtaining all the source and destination IPv4 addresses, MAC address, victim switch DPIDs, and port numbers of the victim switches table-miss packets.
- The flow-rule manager obtains the victim OpenFlow switches flow-rules by using a controller-to-switch message. It compares the values of flow-rules fields with the attack topology. If any of the value of the flow-entry fields matches any of the saturation attack topology values, the flow-entry will be considered as a malicious one. In this case, the flow-rule manager uses the controller-to-switch message to delete the identified malicious flow-entries from the flow-tables of the victim's switch. As a result, a large amount of memory is freed on the switch flow-tables, which allows the new legitimate flow-entries to be installed and returns the switch settings to their pre-attack state.

VII. EXPERIMENTS

In this section, we first evaluate the supervised and semi-supervised classifiers for offline detection of known and unknown saturation attacks using an existing dataset. Then we evaluate the countermeasure in vSwitchDefender.

A. Offline Evaluation of Vicitm Switch Detection

The raw dataset for the offline evaluation is from our prior work [9]. It was created by using a variety of SDN configurations based on Mininet and floodlight controller v1.2. The network topologies included Star, Mesh, Ring, and Tree. The number of OpenFlow switches ranged from 10 to 200 and the number of hosts ranged from 50 to 300. The normal traffic was created by using various traffic generation tools such as Cisco Trex, D-ITG, and Ostinato to mimic the real-world traffic load. For the malicious traffic, the HPING3 hacking tool was employed to lunch TCP-SYN, IP-Spoofing, UDP, ICMP, TCP-SARFU attacks, and combinations of these attacks. The raw dataset is composed of 393 GB benign OpenFlow traffic for a total duration of 237 hours and 150 GB malicious OpenFlow traffic for a total duration of 26.2 hours.

In the prior wok, one minute was found to be the most appropriate time window among many other different sizes studied. It means that each normal and malicious sample in the processed dataset was obtained from one minute of raw OpenFlow traffic. This paper also uses one-minute time window for processing the raw data. Unlike the prior work, however, this paper regroups the OpenFlow traffic with respect to individual OpenFlow switches so as to determine whether each switch is under saturation attack. For each one-minute traffic between the controller and each switch, we calculate the entropy and TPR values, in addition to the total numbers of Packet-In, Packet-Out, Packet-Mod, and TCP-ACK messages. As such, the resultant dataset for evaluating the supervised and semi-supervised classifiers consists of 95,026 normal and 65,148 attack samples.

TABLE I shows the distributions of attack samples. Note that one sample may involve multiple attacks. The traffic data of a single attack sample originated from an attack that involved only one attack type. The traffic data of a multi-attack sample involved more than one types of attacks. For example, a multi-attack sample of UDP includes not only the UDP type, but also one or more of the other attack types.

TABLE I. NUMBERS OF ATTACKS SAMPLES

Attack Type	Single-attack Samples	Multi-attack samples
UDP	2,008	31,012
SYN	2,008	31,012
TCP-SARFU	2,008	37,036
IP-Spoofing	2,007	31,012
ICMP	2,008	33,020

- Detection of known attacks

An attack is considered to be known if it is represented in the dataset used to train a supervised classifier. Because instances of a known attack type are included in the training data, supervised classifiers are likely to classify the attack. This does not apply to semi-supervised classifiers because they are trained with only normal samples. To evaluate the detection of known attacks by a supervised classifier, we

applied stratified 10-fold cross-validation, which divided the 95,026 normal samples and the 6 5,148 attack samples in 10-folds. For each label (normal or attack), all the folds have the same percentage of samples belonging to that label. For each fold, the test dataset contains all the samples of the current fold and the training dataset contains all other folds. The stratification guarantee that for each fold the label distribution is the same.

TABLE II. and TABLE III. show the results of the supervised classifiers using the message header features and the message payloads, respectively. Using the message payloads for classification is slightly better than using the message headers. As shown in TABLE III, their combination has also gained improvement. K-NN is the best among the three supervised classifiers. This shows that the use of both header and the message payloads features together is the best approach, and the unique way to combine them is the use of machine learning. In fact, machine learning automatically learns models to combines all the features for the detection of saturation attacks.

TABLE II. DETECTION OF KNOWN SATURATION ATTACKS-USING OPENFLOW MESSAGE HEADER

Algorithm	Precision	Recall	F1-Score
K-NN	91%	89%	90%
SVM	82%	77%	79%
NB	86%	80%	83%

TABLE III. DETECTION OF KNOWN ATTACKS USING MESSAGE PAYLOAD

Algorithm	Precision	Recall	F1-Score
K-NN	96%	94%	95%
SVM	81%	85%	83%
NB	86%	90%	88%

TABLE IV. DETECTION OF KNOWN ATTACKS USING MESSAGE HEADER AND PLAYLOAD

Algorithm	Precision	Recall	F1-Score
K-NN	98%	95%	96%
SVM	83%	90%	86%
NB	91%	88%	89%

- Detection of unknown attacks with supervised classifiers

An attack is considered unknown if it is not represented in the dataset used to train a classifier. For the five attack types studied in this paper, we first consider each one of them as unknown at a time – the training data includes all other attack types except the target attack type treated as unknown. To evaluate each classifier, we first applied stratified 10-fold cross-validation to all normal and attack samples. Then, for each fold, we created a new training dataset by removing the target unknown attack type and its combinations from the training dataset. Note that we do not remove any attack from the testing dataset. This implies that we test the classifier on the normal traffic samples and all the attack types, including the unknown type. TABLE V. presents the results of classification, where the training data included four types of attacks and the testing data had five types of attacks. It appears that the four types of attacks in the training dataset enable a supervised classifier to generalize the missing one.

To validate this hypothesis, we further evaluated the K-NN (the best performing model in Table V) by including only one attack type in the training dataset, i.e., the stratified 10-fold cross-validation is applied to the 95,026 normal samples and the 2,007/2,008 single-attack samples (Table I). The training data consists of 90% of the normal samples and 90% of the single attack samples; The test data consists of 10% of normal and single-attack samples and all other single and multi-attack samples. TABLE VI. shows that the detection performance has dropped significantly -- using one attack type in the training dataset makes it a challenge for K-NN to generalize for the other four attack types.

- Detection of unknown attacks with semi-supervised classifiers

As semi-supervised classifiers use only normal samples in the training data, all the attack types are considered unknown. We also apply stratified 10-fold cross-validation. For each fold, the training data consists of 90% of all normal samples, and the test data is composed of the rest 10% of the normal samples together with 10% of all attack samples. The results in TABLE VII. show that the semi-supervised classifiers have outperformed K-NN (the best among the three supervised classifiers) in terms of recall and F1-Score. when K-NN was trained with only one type of attack (see Table V). Compared with all the supervised approaches (see tables from II to V), they always provide better recall values even if the precision is sometime 3% lower. Among the four classifiers, variational autoencoder is the most effective in identifying the victim switches when they are targeted by unknown attacks.

TABLE V. DETECTION OF UNKNOWN ATTACKS BY SUPERVISED CLASSIFIERS

Un-known Attack	K-NN			SVM			NB		
	P* %	R* %	F1* %	P%	R %	F1 %	P%	R %	F1 %
UDP	100	94	97	99	90	94	99	92	95
SYN	100	92	96	100	48	65	99	98	98
TCP-SARFU	97	95	96	100	61	76	99	94	96
IP-Spoofing	97	94	95	98	65	78	91	93	92
ICMP	100	54	70	99	47	64	100	53	69

P* = Precision, R* = Recall, and F1* = F1 score

TABLE VI. K-NN DETECTION RESULTS OF UNKNOWN SATURATION ATTACKS USING ONE ATTACK TRAINING DATASET

Known Attack	Precision (%)	Recall (%)	F1 (%)
UDP	96	40	56
SYN	93	17	29
TCP-SARFU	91	30	45
IP-Spoofing	94	23	18
ICMP	93	43	29

TABLE VII. DETECTION OF UNKNOWN ATTACKS BY SEMI-SUPERVISED CLASSIFIERS

Semi-supervised Classifier	Precision (%)	Recall (%)	F1 (%)
Variational Autoencoder	93	98	96
Basic Autoencoder	84	81	82
One-Class SVM	73	75	74
Isolation Forest	82	67	73

B. Online Detection and Countermeasure

vSwitchGuard has been implemented as an application on Floodlight master V1.2 in Python. In principle, the detection module can use any supervised or semi-supervised classifier trained by the offline dataset. The experiment results demonstrate that K-NN is the best among the supervised classifiers for detecting known attacks, whereas variational autoencoder is the best among the semi-supervised classifiers. This suggests that the countermeasure may be based on the results of both K-NN and variational autoencoder for known and unknown attacks: a switch is attacked if either one has reported positive.

To evaluate the effectiveness of the countermeasure, we used Mininet and Floodlight master V1.2 to create simulation SDN environments on a desktop computer equipped with core i5 CPU and 8 GB RAM (similar to the environments used to collect the dataset in the previous section). We have performed 31 case studies. Each includes a period of normal traffic followed by a distinct attack. In other words, the case studies have covered all possible combinations of SYN, IP-Spoofing, UDP, ICMP, TCP-SARFU attacks. In the case studies, we selected different types of network topologies (star, mesh, liners, and tree), network scale, attack type, number of victim switches, and the number of zombie hosts. For the network scales, we considered small networks with 10-20 switches and 20-40 hosts, medium networks with 30-100 switches and 150-300 hosts, and large networks with 110-200 switches and 500-800 hosts.

For each case study, we first ran on each host several normal traffic generation tools without malicious attack. We observed that, in all the 31 cases, the victim switch detection module did not report any attack, and thus the countermeasure module stayed idle. Then we launched the attack for each scenario. On average, the victim switch detection module reported the targeted switches within 60.09 seconds after the attack was launched. This time includes collection of one-minute traffic, feature extraction from the collected traffic, and prediction by the classifier. In some cases, it was shorter than 60.09 seconds. One possible reason is that the attack started in the middle of the time-window and the collected data was sufficient for the classifier to predict the attack. In other cases where it took more than 60.09 seconds, the attack started in the middle of the time-window and the collected traffic was insufficient for the classifier to predict the attack. Thus, vSwitchGuard needs another minute to collect and analyze the traffic. In the case studies, we did not observe any false alarms during the normal traffic. False alarms, however, are possible when the normal traffic becomes similar to the attack traffic. These false alarms will trigger the Packet-In deep inspection filter. Once an attack is reported by the detection module, the countermeasure module will be triggered. In the 31 case studies, the countermeasure model took about 2~7 seconds to recover the targeted switches.

In the following, we use one case study to discuss the performance overhead of vSwitchGuard in terms of CPU utilization, OpenFlow channel bandwidth utilization, and flow table utilization. The SDN network was based on a ring topology with 15 switches. Five zombie hosts were used to launched a UDP attack that targeted three switches. We measured the CPU utilization using NetData, before, during and after the attack. Figure 4 shows the changes of CPU utilization. Before the attack (from 0–58 second), the average CPU utilization was about 45%-50% since the SDN environment and the controller were running on the same machine. During the attack (from 59 – 119 second), the CPU utilization reached around 95%. Meanwhile, the traffic collector and feature extractor collects were working. At second 120.30, the CPU utilization went down quickly (around 45%) because vSwitchGuard had identified the targeted switches and mitigated the attack. The total time for detection and countermeasure was about 2.2 seconds.

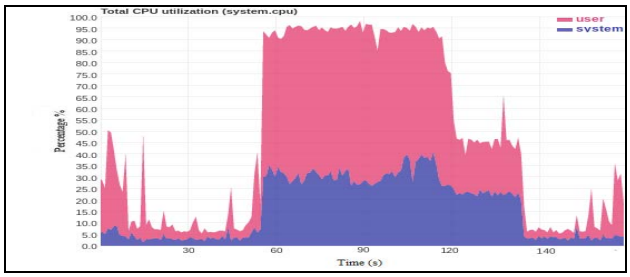

Figure 4. CPU Utilization under UDP Attack.

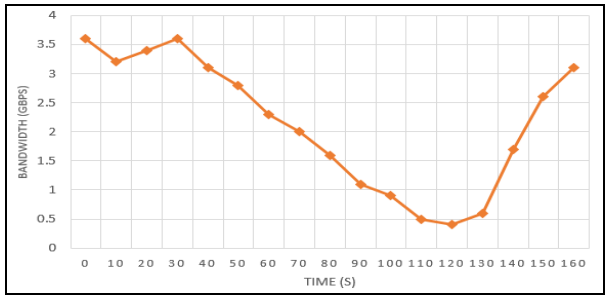

Figure 5. OpenFlow Channel Bandwidth Utilization under UDP Attack.

We also measured the bandwidth of the OpenFlow channel before, during, and after the attack. The result is shown in Figure 5. The average bandwidth of the OpenFlow channel before the attack was about 3.2 GBPS. After the attack started, the bandwidth decreased to 0.43 GBPS. After vSwitchGuard had handled the attack, the bandwidth started peaking up slowly due to the huge amount of Packet-Out

and Packet-Mod messages forwarded from the controller to the victim switch. This demonstrates that vSwitchGuard had defended the SDN network against the attack without significant performance overhead.

TABLE VIII. presents flow-table utilization of the victim switches, in comparison with FloodGuard and FloodDefender. vSwitchGuard did not overload the network when there was no saturation attack. When the attack occurred, the flow-table utilization of victim switches protected by vSwitchGuard remained steady since all the malicious flow-rules were removed from the victim switches. The total flow-table utilization rate caused by vSwitchGuard was about 1% due to the installation of the blocking flow rules. For FloodDefender, the flow-table utilization can reach up to 15% of the flow-table buffer due to the installation of monitoring and processing flow-rules. For FloodGuard, the flow-table utilization can reach up to 30% since it uses rate control to protect the controller and switches. As such, vSwitchGuard provides a more efficient way to handle malicious table-miss packets with lower overhead of flow-table utilization.

TABLE VIII. FLOW-TABLE UTILIZATION UNDER UDP ATTACK

	vSwitchGuard	FloodDefender	FloodGuard
No Attack	4% ~ 5%	4% ~ 5%	4% ~ 5%
Under Attack	5% ~ 6%	19% ~ 20%	34%~35%

VIII. CONCLUSION

We have presented vSwitchGuard as an effective framework for defending OpenFlow switches against saturation attacks. The experiments have evaluated three supervised and four semi-supervised classifiers for detecting known and unknown attacks. The countermeasure can effectively recover the victim switches. Compared to the existing work, vSwitchGuard incurs lower performance overhead in terms of CPU utilization, OpenFlow channel bandwidth, and flow-table utilization.

This paper has studied five types of saturation attacks in SDN environments. Our future work will investigate more types of attacks and evaluate the effectiveness of supervised and semi-supervised classifiers.

REFERENCES

[1] https://www.opennetworking.org/

[2] S. Shin, V. Yegneswaran, P. Porras, and G. Gu. "Avant-guard: Scalable and vigilant switch flow management in software-defined networks." in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 413-424. ACM, 2013.

[3] R. Mohammadi, R. Javidan, and M. Conti, "Slicots: an sdn-based lightweight countermeasure for tcp syn flooding attacks," *IEEE Transac- tions on Network and Service Management*, vol. 14, no. 2, pp. 487–497, 2017.

[4] M. Ambrosin, M. Conti, F. DeGaspari, and R. Poovendran,"Lineswitch: tackling control plane saturation attacks in software-defined networking," *IEEE/ACM Transactions on Networking*, vol. 25, no. 2, pp. 1206–1219, 2017.

[5] H. Wang, L. Xu, and G. Gu, "Floodguard: A dos attack prevention extension in software-defined networks," in *Proceedings of the 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2015, pp. 239–250.

[6] G. Shang, P. Zhe, X. Bin, H. Aiqun, and R. Kui, "Flooddefender: Protecting data and control plane resources under sdn-aimed dos attacks," in *Proceedings of the IEEE International Conference on Computer Communications(INFOCOM)*. IEEE, 2017, pp. 1-9.

[7] Z. Li, W. Xing, S. Khamaiseh, and D. Xu, "Detecting saturation attacksbased on self-similarity of openflow traffic,"IEEE Transactions on Network and Service Management, pp. 1–1, 2019.

[8] D. Hu, P. Hong and Y. Chen, "Fadm: Ddos flooding attack detection and mitigation system in software-defined networking," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*. 2017, pp. 1-7.

[9] S. Khamaiseh, E. Serra, Z. Li, and D. Xu, "Detecting saturation attacks in sdn via machine learning," in 2019 4th International Conference on Computing, Communications and Security (ICCCS). IEEE, 2019, pp. 1–8.

[10] P. W. Chi, C. T. Kuo, J.W. Guo, and C. L., Lei, "How to detect a compromised sdn switch," in *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 2015, pp. 1-6.

[11] H. Zhou, C. Wu, C. Yang, P. Wang, Q. Yang, Z. Lu, and Q. Cheng, "Sdn-rdcd: a real-time and reliable method for detecting compromised sdn devices," *IEEE/ACM Transactions on Networking (TON),* vol. 26, no. 5, pp. 2048-2061, 2018.

[12] R. Swami, M. Dave, and V. Ranga, "Software-defined networking-based ddos defense mechanisms," *ACM Computing Surveys (CSUR),* vol. 52, no. 2, p. 28, 2019.

[13] J. Asharf and S. Latif, "Handling intrusion and ddos attacks in software defined networks using machine learning techniques," in *2014 National Software Engineering Conference*. IEEE, 2014, pp. 55-60.

[14] Q. Niyaz, W. Sun, and A. Javaid, "A deep learning based ddos detection system in software-defined networking (sdn)," *arXiv preprint arXiv:1611.07400,* 2016.

[15] R. Braga, E. Mota, and A. Passito, "Lightweight ddos flooding attack detection using nox/openflow," in *LCN*, vol. 10, 2010, pp. 408-415.

[16] A. Abubakar and B. Pranggono, "Machine learning based intrusion detection system for software defined networks," in *2017 Seventh International Conference on Emerging Security Technologies (EST)*. IEEE, 2017, pp. 138-143.

[17] T. A. Tang, L. Mhamdi, D. McLernon, S. A. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, IEEE, 2016, pp. 258-263

[18] D. Santos, J. Wickboldt, L. Granville, and A. Schaeffer-Filho, "Atlantic: A framework for anomaly traffic detection, classification, and mitigation in sdn," in *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2016, pp. 27-35.

[19] J. Ye, X. Cheng, J. Zhu, L. Feng, and L. Song, "A ddos attack detection method based on svm in software defined network," *Security and Communication Networks*, vol. 2018, 2018.