Detecting Saturation Attacks in SDN via Machine Learning

Samer Khamaiseh, Edoardo Serra
Department of Computer Science
Boise State University
Boise, ID 83725, USA
{samerkhamaiseh,edoardoserra@boisestate.edu

Zhiyuan Li
Dept. of Cybersecurity
Jiangsu University
Jiangsu 212013, China
lizhiyuan@ujs.edu.cn

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

Abstract— **Software Defined Networking (SDN) is a new network paradigm that facilitates network management by separating the control plane from the data plane. Studies have shown that an SDN may experience a high packet loss rate and a long delay in forwarding messages when the OpenFlow channel is overwhelmed by a saturation attack. The existing approaches have focused on the detection of saturation attacks caused by TCP-SYN flooding through periodic analysis of network traffic. However, there are two issues. First, previous approaches are incapable of detecting other types, especially unknown types, of saturation attacks. Second, they rely on pre-determined time-window of network traffic and thus are unable to determine what time window of traffic data would be appropriate for effective attack detection. To tackle these problems, this paper first investigates the impact of different time-windows of OpenFlow traffic on the detection performance of three classification algorithms: the Support Vector Machine (SVM), the Naïve Bayes (NB) classifier, and the K-Nearest Neighbors (K-NN) classifier. We have built and analyzed a total of 150 models on OpenFlow traffic datasets generated from both physical and simulated SDN environments. The experiment results show that the chosen time-interval of OpenFlow traffic heavily influences the detection performance -- larger time-windows may result in decreased detection performance. In addition, we were able to achieve reasonable accuracy on detection of unknown attacks by applying proper time-windows of OpenFlow traffic.**

Keywords— Software-defined networking, OpenFlow, saturation attack, anomaly detection, machine learning

I. Introduction

As an emerging network architecture, SDN suffers from many security threats. Kreutz et al. [1] discussed several security issues, including flooding of the data and control planes, vulnerabilities of OpenFlow switches, and the lack of authorization. One particular challenge is dealing with the Distributed Denial of Service (DDoS) attacks [2]. Recent studies have focused on detecting and mitigating DDoS saturation attacks, especially control plane saturation attack via SYN flooding attacks [4][5][6][7]. The limited bandwidth of OpenFlow communication channel can become a bottleneck for the entire network. A hacker may exploit this limitation by launching a flooding attack. Such saturation attacks range in duration, and a long-lasting attack can affect the entire SDN. Most of the existing machine learning detection methods rely on an arbitrary predefined, fixed time-window to start analyzing the network traffic to detect the saturation attacks [3][4][5][6][7]. If the time-window is too large, the detection method response time will be long, and the attack may have an opportunity to saturate the network. If the time-window is too small, the amount of traffic may be insufficient to provide reliable detection results. In this case, the detection method can also become a huge performance-overhead for the SDN. Thus, identifying the proper time-window of traffic for effective detection is a key concern.

Machine learning has been successfully applied to various security problems. For detection of saturation attacks, however, the adoption of machine learning in the "real-world" has been very limited. This is partly because of their deficiencies for dealing with unknown attacks. An unknown attack is an attack which is not represented in the dataset used to train the detection model [8]. Because there are no instances of the attack included in the training set, supervised machine-learning methods are unable to classify it. In this paper, we aim to answer the following questions: (1) Is it possible to determine a proper time-window for conducting OpenFlow traffic analysis, in order to provide the best results for detection of saturation attacks? (2) How does the variation in the time-windows affect the detection performance of a machine learning classifier? (3) are supervised machine learning algorithms effective in detecting unknown attacks?

For the first two questions, we have evaluated the detection performance of three state-of-the-art machine learning algorithms: the widely-used Support Vector Machine (SVM), K-Nearest Neighbor (K-NN) classifier, and Naïve-Bayes (NB) classifier. We used a variety of time-windows of OpenFlow traffic to determine the proper time-window for detection of saturation attacks. Also, we studied the impact of different time-windows of OpenFlow traffic on the classifiers' detection performance by conducting a false-negative analysis. We conducted extensive experiments using both physical and simulated SDN environments. In each environment, we collected "normal" OpenFlow traffic by using different traffic generation tools that mimic real-world network traffic with various network protocols and internet applications. We collected malicious OpenFlow traffic by performing 31 saturation attacks, combining UDP flooding, SYN flooding, IP Spoofing, ICMP, and SARFU flooding. In both environments, we extracted datasets representing time-windows ranging from 1-minute of OpenFlow traffic analysis to the entire duration. We performed a total of 150 experiments to evaluate the impact of different time-windows of OpenFlow traffic on the performance of K-NN, NB, and SVM. This allowed us to identify the earliest time-window for detecting saturation attacks with the highest detection performance. The detection performance may be decreased when the time-window of OpenFlow traffic become larger.

For the third question, we evaluated SVM, K-NN, and NB for detecting unknown saturation attacks. To test a supervised classifier, we excluded the observations related to one type of a saturation attack from the training dataset, to act as an unknown attack for the models. The test dataset includes the

978-1-7281-0875-9/19/$31.00 ©2019 IEEE

observations from the unknown attacks and a random set of normal traffic observations. Then we used our evaluation metrics to assess the classifiers for detecting the unknown attacks. Our experiment results showed that SVM, NB, and K-NN are capable of detecting unknown saturation attacks.

The contributions of this paper are as follows:

- To the best of our knowledge, this work is among the first to investigate the impact of different time-windows of OpenFlow traffic on the performance of supervised classifiers for the detection of saturation attacks in SDNs.
- This work has created two large saturation attack datasets by using both physical and simulated SDN environments. This can further promote the research on SDN security.
- This is the first work on the evaluation of supervised classifiers for detecting unknown saturation attacks in SDN environments.

The remainder of this paper is organized as follows. Section II presents a brief introduction to the SDN, OpenFlow protocol, and SDN saturation attacks. Section III reviews the related work. Section IV describes the methodology. Section V presents our experiments. Section VI concludes the paper.

II. BACKGROUND

A. *SDN and OpenFlow Protocol*

In traditional computer networks, the administrator needs to configure the network devices to change the route of the packets because the control is distributed among all the network devices. SDN offers a new way of managing and controlling networks by separating the control plane and data plane. The basic architecture of an SDN environment is composed of data plane and control plane that communicate through the southbound API. The data plane includes the network hardware components in term of switches (e.g. OpenFlow switches) and routers responsible for forwarding operations. The SDN hardware components do not need to be changed over time to upgrade the network because they are programmable, and they are controlled by the SDN controller via southbound API. The southbound API represents the interface between the network switches and the SDN controller. It allows the controller to control the behavior of hardware devices. The OpenFlow protocol is the standardized and the most widely-used southbound API.

The SDN controller is a centralized controlling unit that translates the SDN applications network requirements down to the data plane and provides the business application that resides in the application layer with the network information such as network topology and statistical reports. The communication between the business applications and the SDN controller is through the northbound APIs. The northbound APIs provide an abstraction of the network functions and enable the network applications and orchestration systems to dictate the behavior of the SDN network by providing a programmable interface to request the network services and dynamically configure the network.

OpenFlow as a communication protocol between the data plane and the control plane has been defined as the standard southbound APIs used in the SDN architecture by Open Network Foundation (ONC). An OpenFlow switch comprises of flow tables, group tables and OpenFlow channel that provide the connection channel to exchange the OpenFlow messages between the SDN controller and OpenFlow switches.

B. *SDN Saturation Attacks*

When a new incoming packet does not match any of the existing flow-entries of the OpenFlow switch, a table-miss occurred. The data plane encapsulates the new packet insides a Packet-In message and sends it to the SDN controller for deciding the fate of the table-miss packet. The feature of table-miss can be exploited by a hacker to consume the computation resource (e.g., CPU and memory) of the controller, switches and saturate the OpenFlow communication channel. A hacker can lunch data-to-control plane saturation attack by triggering a huge number of table-miss packets by sending a vast number of spoofed packets to reduce the possibilities of matching any of the existence flow-entries on the targeted switch. The controller will be overwhelmed dealing with the Packet-In messages. Such data-to-control plane flooding attack may exhaust the controller computation resources and delay the forwarding messages to the OpenFlow switches.

When the data-to-control plane occurred, the controller will trigger a large amount of Packet-Out and Packet-Mod messages, which cause a control-to-switch flooding attack. As a result, numerous invalid flow-rules will be installed in the targeted switches flow-tables. This prevents the valid flow-entry to be installed, which in turn causes the benign packets to be dropped. Therefore, all the benign packets will be rejected by the OpenFlow switches and the OpenFlow channel will be saturated.

III. RELATED WORKS

Studies have shown that there are various security threats to SDN [1]. For example, flooding attacks that disturb the SDN-based network can put it out-of-service. Several research works have proposed ways to detect, mitigate, and prevent the SYN-Flooding attack using machine learning and deep learning approaches. Ashraf et al [9] have investigated various machine learning algorithms that can be used to detect DDoS attacks in the SDN environment. Quamar et al. [10] proposed an SDN network application that adapted a stacked autoencoder (SAE) deep learning technique for detecting multi-vector DDoS. The proposed 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 instead of flow sampling, which requires extensive computational resources. The dataset used for training and testing the proposed system was collected from a traditional wireless-network, not an SDN-based network. Ahmed et al. [7] proposed a DDoS detection prototype by using Dirichlet process mixture model to detect the attack traffic. Its misclassification rate (for attack traffic versus benign traffic) is around 50%. Hu et al. [11] proposed the FADM framework for detecting and mitigating SYN, ICMP, and UDP flooding attacks in SDN. FADM has two components: (1) a DDoS detection module which uses the sFlow approach to collect the network information, extracts features using the entropy approach, and detects the attack traffic using an SVM algorithm, and (2) a DDoS mitigation module that is responsible for mitigating the DDoS attacks using a white-list and traffic migration approach. The use of the SVM classifier required very long training and prediction time to distinguish the normal and attack traffic. Braga at al [12] proposed a lightweight DDoS flooding attack detection in SDN. They used a self-organized map (SOM) to detect the attack traffic. The shortcoming of this approach is the training and detection time required for the SOM algorithm, which is around a few hours for training and a few minutes for classifying the traffic.

Tuan et al. [13] used the Deep Neural Network (DNN) to develop an anomaly DoS detection system. The accuracy is 88.04%, which is relatively low. The NSL-KDD dataset, generated from a traditional network, is used for training and testing the detection model. Abubakar and Pranggono [14] developed a flow-based anomaly detection system using a neural network. The NSL-KDD dataset was used to train and evaluate the model. Shine et al [15] proposed the AVANT-GUARD framework to mitigate the TCP-SYN flooding that is sent to the SDN controller, 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 pre-defined threshold, which cannot accurately differentiate between the normal and abnormal SYN packets.

Wang et al. [5] proposed Floodguard: a prevention approach against DoS attacks. FloodGaurd acts as middleware between the controller and its applications and has three components: a detection module, a flow rule analyzer module, and a packet migration module. The FloodGaurd detection module monitors the OFPT_PACKET_IN messages and triggers the other modules when the OFPT_PACKET_IN messages exceed the pre-defined thresholds. Shang et al. [6] introduced FloodDefender as a framework for mitigating DoS attacks against the SDN network. It has four modules: attack detection, table-miss engineering, packet filter, and flow rule management. The FloodDefender detection module uses the same detection technique in FloodGuard. It uses the SVM to classify the malicious packets and normal packets, which require long training time. The main limitation of the detection module in both proposed approaches is the fact that using the OPPT_PACKET_IN message rate to detect the attacks is not accurate since heavy normal traffic can generate similar OFPT_PACKET_IN messages. Mousavi and Hilaire [19] proposed an early detection approach for DDoS attacks using the entropy values of the IP addresses for new incoming packets to the controller, within the first 250 Packets. However, the major drawback of this approach is based on the assumption that each new packet forwarded to the controller is a malicious packet if the source addresses match any of the existing network hosts. In addition, if the source IPv4 address appears in many packets, the entropy value will be lower than the predefined thresholds, which means that the host is under attack. This approach can generate many false alarms of early attack detection, specifically, if the SDN network is in a reactive flow-management configuration. This means that the flow-entries will be configured dynamically to provide a flexible way to control the network traffic. Thus, in a large-size SDN network, many legitimate packets forwarded to the controller and many legitimate flow-entries will be installed to control network traffic. However, the proposed approach cannot be utilized as an early DDoS attack detection method since the normal behavior of the SDN network will always be considered as malicious behavior. Zhang et al [16] introduced two novel attacks: a table-miss striking attack and a counter manipulation attack. They provided the SWGuard system as a solution to detect and prevent these attacks.

The existing works typically use a pre-defined time-window, without justification about the specified time-window. This research aims to determine the proper time-window to detect saturation attacks and show the impact of different time-windows of OpenFlow traffic on the detection performance of the K-NN, SVM, and NB classifiers. This research also investigates the capabilities of these classifiers for detecting unknown saturation attacks, which is considered a huge obstacle to adopting classifier-based detection methods to real-world SDN networks [8].

IV. METHODOLOGY

In this section, we first introduce the features that we extracted from the OpenFlow traffic and explain our approach for preprocessing the OpenFlow traffic to generate multiple datasets for different time-windows. We also introduce the classifiers for detecting the known and unknown saturation attacks and describe the experimental setup and data collection of OpenFlow traffic.

A. Feature Extraction and Data Preprocessing

The OpenFlow traffic [17] is a sequence of packets that are transferred between the controller and the switch. Each packet has different attributes such as the packet time, the source and the destination IP addresses, the OpenFlow message type, and the length of the packet. Formally, the OpenFlow traffic OF is a sequence of OpenFlow packets $< p_1, p_2 \ldots, p_n >$ captured during a normal or attack session, where each packet p_i has $<time,\ srcIP,\ dstIP,\ OF\, msg,\ length>$. The OpenFlow traffic consists of 29 types of OpenFlow messages that can be categorized into three main types: (1) controller-to-switch messages that are sent from the controller to the switch to acquire information and modify the switch state (e.g., Packet-out, Packet-mod, and Role-request), (2) asynchronous messages that are sent by the switch to the controller to inform about new coming packets, error, and switch state changes (e.g., Packet-in, Flow-removed, Port-status, and error), and (3) symmetric messages that are sent in both sides such as Hello and Echo messages.

We consider the following four features of the captured OpenFlow traffic:

- The number of OFPT_PACKET_IN messages sent from the switch to the controller.
- The number of OFPT_PACKET_OUT messages sent from the controller to the switch.
- The number of OFPT_PACKET_MOD messages sent from the controller to the switch.
- The number of TCP_ACK messages sent from the switch to the controller, or vice versa.

These features are selected based on our analysis and observation of the OpenFlow traffic behavior in physical and simulated SDN environments when they are in attack mode or normal mode. The features are sensitive to various saturation attacks and their combinations. Fig. 1 shows the impact of a UDP saturation attack on the Packet-In and Packet-Out features. When the UDP flooding attack occurs, the attacking host tries to flood the SDN network by generating a massive amount of IP packets including UDP datagrams. Meanwhile, the OFPT_PACKET_IN messages generated by UDP attack come from the switch that is connected to the attacking host. Thus, the number of OFPT_PACKET_IN messages in the OpenFlow traffic increases significantly, and the number of OFPT_PACKET_OUT messages decreases significantly. TABLE I. summarizes the changes to each feature caused by UDP, SYN, ICMP, IP Spoofing, and SARFU TCP saturation attacks.

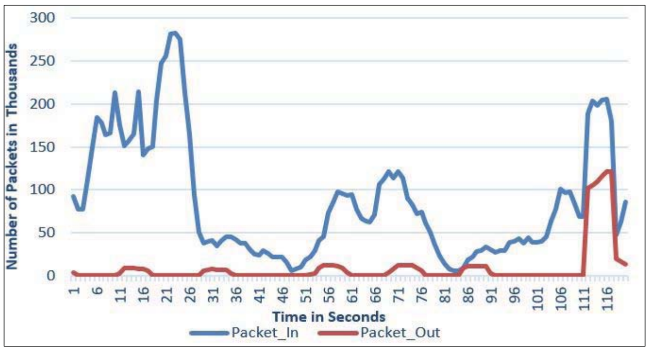

Fig. 1 Effect of UDP Flooding Attack on Packet-In & Packet-Out Messages

Algorithm 1: OpenFlow Dataset Generation

```
Input      OpenFlow traffic OF, Time-Window T, Time-Shifting S,
           Traffic-Type L {0,1}
Output     Dataset $X_J$
Declare    packetIn, packetOut, packetMod, tcpAck, $Msg_{(type)}$
Steps
1     J=0 (the index of the output sample $X_J$)
2     Repeat:
3        firstPacketIndex=1 ($p_i$ is the first packet of the current shift)
         $startTime = p_{time_{(firstPacketIndex)}}$ (is the first packet time of the
         current shift)
4        for ( i = firstPacketIndex; i < n; i++) do
5            if $p_{time} - startTime > T$
6               createNewSample_Xj(packetIn,packetOut,tcpAck,J++)
                addNewSampleXj←xj with corresponding traffic type L
               and increase J by one.
7              break;
8            Endif
9            switch ($Msg_{(type)}$) {
10            Case1: $Msg_{(type)}$ = "OFPT_PACKET_IN"
11                    packetIn += 1;
12            Case2: $Msg_{(type)}$ = "OFPT_PACKET_OUT"
13                    packetOut += 1;
14             Case3 : $Msg_{(type)}$ = "OFPT_PACKET_MOD"
15                     packetMod += 1;
16             Case4: $Msg_{(type)}$ = "TCP_ACK"
17                    tcpAck += 1;
18           }
19       end for
20    firstPacketIndex=updatePacketIndexNextShfit(firstPacketIndex,S)
21    Until firstPacketIndex > n
```

To investigate the appropriate time-window for detecting saturation attacks, we tested different time windows and evaluated their impact on the detection performance of the SVM, K-NN, and NB classifiers. From each time window, a different dataset was generated from the collected OpenFlow traffic in both physical and simulated SDN environments. The time-windows ranged from one minute to the entire attack duration. A detailed description of our approach for extracting these datasets from the OpenFlow traffic is given in Algorithm 1. The features collected in each dataset were the OpenFlow traffic session OF, the dataset time-window T, the dataset time-shifting S, and the OpenFlow traffic type L which is $< L \leftarrow 0 \; if \; OF \; is \; normal \; or \; L \leftarrow 1 \; if \; OF \; is \; attack >$. The output was a dataset $X_J = (x_1, x_2, x_3 \ldots, x_n)$ which was a sequence of labeled samples, with each sample x_j in the form of <*number of Packet_in messages , number of Packet_out messages, number of Packet_mod messages, number of TCP_ACK message* >. Lines (9-18) deal with extracting our features from the OpenFlow traffic packet sequence *OF* for the specified time-window *T*. For each packet, we extracted the OpenFlow message type. If the message type matched any of our messages, the corresponding counter increased by one (lines 10-17). When the difference between the packet time and the starting time is larger than *T*, a new sample x_j created with the corresponding label and increase the dataset index J by one (lines 5-8). After each new sample x_j, the value of the $firstPacketIndex$ updated by adding the shifting parameter *S* for the next shift starting index (line 20). The reason behind including the shifting parameter was to increase the overlapping in the generated datasets.

TABLE I. IMPACTS OF SATURATION ATTACKS ON THE KEY OPENFLOW MESSAGES

Saturation Attacks	**The Impact**			
	# Packet-In	***# Packet-Out***	***# Packet-Mod***	***# TCP-ACK***
UDP	Significant increase followed by a decrease Significant decrease	Significant decrease	Significant decrease	Significant increase
SYN	Significant increase	Increase and then a significant decrease	Increase and then a significant decrease	Significant increase
ICMP	Insignificant increase	Increase for a short period of time followed by a significant decrease	Increase for a short period of time followed by a significant decrease	Noticeable increase followed by a significant decrease
IP Spoofing	Increase followed by a significant decrease	Increase followed by a significant decrease	Increase followed by a significant decrease	Increase and significant decrease to be a zero packet
SARFU TCP	Increase and then noticeable decrease and then decrease to zero packets.	Increase and then significant decrease to zero packets.	Increase and then noticeable decrease and then a significant decrease to zero packets	Increase and then noticeable decrease

B. Classifiers

We train several supervised models by using the obtained datasets *Xj* as explained in the previous section and evaluate their performance. In this paper, we consider K-Nearest Neighbor Classifier (K-NN) a well-known supervised learning algorithm that has been widely used in intrusion detection. K-NN is a non-parametric learning algorithm which stores all the instances of the training dataset. K-NN is also a lazy learning technique that does not use the training points for making any generalizations. Another machine-learning algorithm which has proven to be popular for abnormal traffic detection is *Naïve-Bayes (NB).* The NB classification model uses the Bayesian theory that includes

Bayesian learner and statistical analysis. We also consider Support Vector Machine (SVM), another well-known supervised machine learning algorithm that has been used widely in different fields. Recently, several studies have used SVM in SDN-intrusion detection systems and in traditional network intrusion detection systems [18]. SVM can be used for classification and regression analysis. It works by plotting the observations as points in N-dimensional space (where N is the number of features) and outputting an optimal hyperplane that maximizes the margin between two classes. This boundary is then used to categorize new observations.

C. Experiment Setup and OpenFlow Traffic Collection

We collected the OpenFlow communication channel traffic using both physical and simulated environments. The main advantage of using a physical SDN environment is the capacity to replicate the workload of a real-world SDN network and the internet traffic generated by real-world applications. The physical SDN environment consists of a Pica8 P-3290 OpenFlow switch, Floodlight Master 1.2v as an SDN controller, and five hosts ranked from h-1 to h-5. 0shows the physical environment configurations. The physical environment is limited to the network scale and topology. The simulated SDN environments was created using the Mininet tool [21] with different network topologies (i.e., tree topology, star topology, mesh topology, and linear topology), and different network scale (i.e., number of hosts, number of switches). TABLE III. shows the main configurations of the simulation SDN network.

The benign OpenFlow traffic was collected from both physical and simulated environments by using four traffic generation tools that mimic real-world network behaviors. We used D-ITG (Distributed Internet Traffic Generator) [22]. D-ITG has ITGSend and ITGRecv components. ITGSend can generate parallel traffic flows and send it to different ITGRecv instances. ITGRecv is responsible for receiving traffic flows from ITGSend. D-ITG provides the ability to generate multiple unidirectional traffic flows for different protocols such as IPv4, IPv6, TCP, UDP, ICMP, SCTP (Stream Control Transmission Protocol), DCCP (Datagram Congestion Control Protocol), DNS, Telnet, and VoIP. We used the Nping open source tool [25] that can generate traffic for different protocols such as the ARP protocol. By using Nping, we were able to customize the packet size and the transmission intervals of the generated traffic. In order to generate a concurrent stateful and stateless traffic that simulated the internet traffic, we used Cisco's TRex realistic traffic generator [23]. TRex can generate almost any kind of L4-7 traffic, based on the smart reply of real traffic templates. It can amplify the traffic of the server and the client side up to 200Gb/sec. We also exploited OSTINATO [24] to configure and generate many traffic streams for different protocols such VLAN, IPv4, IPv6, stateless TCP, ARP, ICMPv4, ICMPv6, IGMP (Internet Group Management Protocol), MLD (Multicast Listener Discover), RTSP (Real Time Streaming Protocol) and NNTP (Network News Transfer Protocol). The total size of the captured benign OpenFlow traffic from the physical environment was 250 GB, the total duration was about 137 hours, and the total simulated benign traffic was about 143 GB, for a total duration of 100 hours.

In both physical and simulated environments, Hping3 [27] and LOIC (Low Orbit Ion Canon) [26] were employed to lunch the saturation attacks. In each environment, 31 saturation attacks were performed to cover all combinations of SYN flooding, UDP flooding, ICMP, IP Spoofing, and SARFU-TCP flooding. Each of the attacks flooded the control and/or data planes. The total size of the physical environment malicious traffic was 50Gb and the duration for each attack was about 30 minutes. For the simulated environment, the total size of the malicious traffic was about 100 Gb and the duration of each attack was about 20 minutes.

TABLE II. PHYSICAL ENVIRONMENT CONFIGURATIONS

Host Name	CPU Info	Memory Info	Operating System
Controller Machine	Intel Core (i7) 2.5GHz	16GB	Ubuntu 16.04.5 LTS
h-1	Intel Core (i5) 2.5GHz	8GB	Ubuntu 16.04.5 LTS
h-2	Intel Core (i5) 2.5GHz	8GB	Ubuntu 16.04.5 LTS
h-3	Xenon E5 2.5GHz	4GB	Ubuntu 16.04.5 LTS
h-4	Xenon E5 2.5GHz	4GB	Ubuntu 16.04.5 LTS
h-5	Intel Core (i5) 2.4GHz	8GB	Ubuntu 16.04.5 LTS

TABLE III. SDN SIMULATION ENVIRONMENTS CONFIGURATIONS

Parameter	Description	Default Value
NC	Number of Controllers	1
NS	Number of switches	10-200
NH	Number of Hosts	50-300
NT	Network Topology	Star, Mesh, Ring, Tree

V. EXPERIMENT RESULTS AND DISCUSSIONS

This research aimed to answer the following questions:

- RQ1: What is a proper time-window of OpenFlow traffic analysis for detecting saturation attacks?
- RQ2: How do different time-windows affect the detection performance of a classifier?
- RQ3: Are classifiers effective in the detection of unknown saturation attacks?

A. Proper Time-Window for Detection of Known Attacks

In this section, we describe the experiments we conducted to discover the proper time-window for OpenFlow traffic analysis to early detect the saturation network attacks. Also, we evaluate the impact of different time-windows of OpenFlow traffic analysis on the detection performance of the known saturation attacks for SVM, K-NN, and NB classifiers. We refer to known saturation attack as an attack that has been included in the training phase, in other words, the training dataset has many samples that describe the attack behavior and our classifier models have been trained to detect this attack. In the physical environment, 30 datasets are generated from the collected OpenFlow traffic and each dataset represents a different time-window, range from 1 minute to the attack duration on physical environment that's equal to 30 minutes. In our experiments, each dataset is used to train and test K-NN, SVM, NB models and we used the precision, recall, and F1 score metrics to evaluate the performance of our models and analyze the impact of different time-window on the model's prediction results. We now discuss how we determined the earliest proper time-window for each classifier, from these experimental results.

Fig. 2 shows the precision, recall, and F1 score metrics for the K-NN models when the time-window ranges from 1 minute to 30 minutes (each minute represents a trained K-NN model). The highest detection rate for the K-NN classifier is obtained when the time-window is equal to 1 minute: the precision is 96% with recall 95%, and the F1 score is 95%. The lowest detection rate is when the time-window equal 30-minutes, in which case, the corresponding precision is 47% with recall 98%, and the F1 score is 64%. As a result, in the physical environment, the proper time-window for the K-NN classifier is 1 minute of traffic analysis. To support this conclusion, Fig. 2 shows that the precision and F-1 score decrease as the time-window of traffic analysis increases.

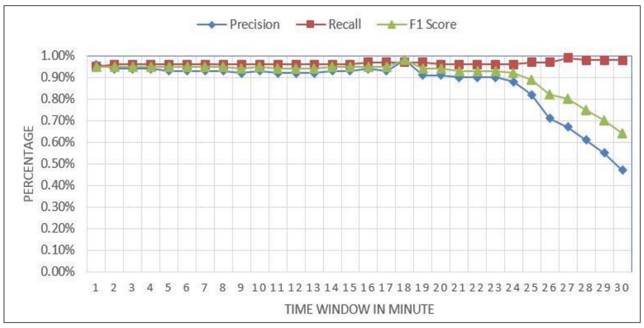

Fig. 2 K-NN on the Physical Environment

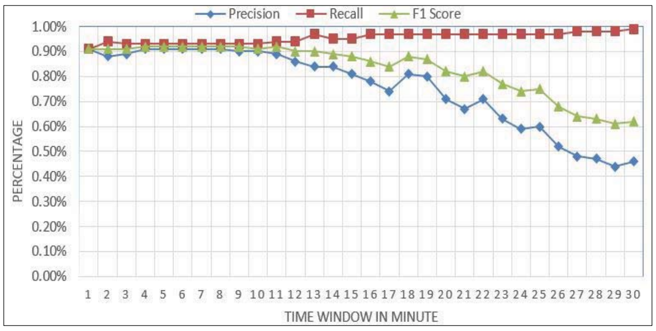

Fig. 3 SVM on the Physical Environment

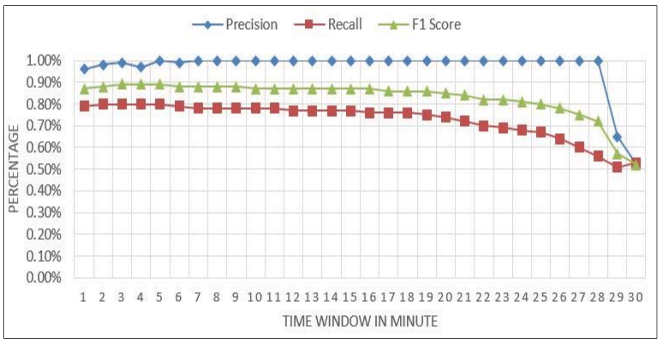

Fig. 4 NB on the Physical Environment

Fig. 3 shows the values of the evaluation metrics for the SVM models. Each time-window represents a trained SVM model. The highest detection result is when the time-window equals 1 minute; in this case, the precision is 91% with a recall of 91%, and the F1 score is 91%. In contrast, the lowest precision achieved by these models is 46% with a recall of 99%, and F1 score of 62%. This is when the time-window is 30 minutes. Fig. 3 shows the impact of time-window length on the SVM classifier performance. Notably, the precision and F1 score decrease and the recall increase as the time-window increases. Thus, the proper time-window for the SVM classifier in the physical environment is also 1 minute.

Similar to the approach for the K-NN and SVM classifiers, we performed 30 experiments to evaluate the NB classifier detection performance for the same time-windows. Fig. 4 shows the results of our evaluation. The highest precision is 99% with a recall of 80%, and an F1 score of 89%, when the time-window equals 3 minutes. The lowest precision is 52% with a recall of 53%, and an F1 score of 52% when the time-window equals 30 minutes. In the physical environment, the 3-minute time-window seems proper for the NB classifier in order to obtain a high detection rate.

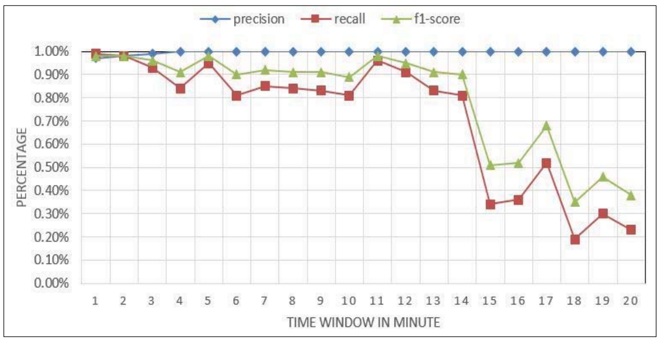

Fig. 5 K-NN on the Simulation Environment

In the simulation environment, 20 datasets were generated and used in training and testing the proposed classifiers. Similar to the physical environment experiments, each dataset represented a different time-window of traffic analysis. In this case, the dataset time-windows ranged from 1 minute to 20 minutes (i.e., attack duration on simulation environments equal to 20 minutes). Again, we used the precision, recall, and F1 score to evaluate the performance of our classifiers in each experiment. Fig. 5 shows that when the time-window equals 1 minute, the K-NN classifier achieves a high precision of 97% with a recall of 99% and an F-1 score of 98%. When the time-window equals 18 minutes, the K-NN classifier achieves the highest precision overall (100%) but suffers from low recall (19%). The F1 score is 35%. As a result, we conclude that the 1-minute time-window is proper for the K-NN classifier, in order to obtain the highest detection results in the simulation environment. Fig. 5 shows the impact of the time-window on the detection performance of the K-NN classifier. Increasing the time-window led to increased precision but decreased the recall and the F1 score ratios.

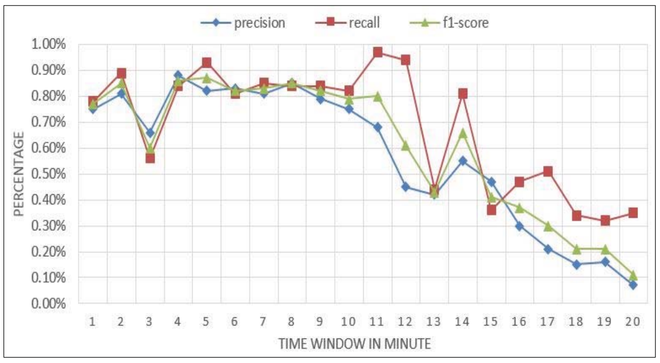

Fig. 6 SVM on the Simulation Environment

As shown in Fig. 6, the SVM classifier achieved the highest detection results with a time-window of 2 minutes. The corresponding precision is 81% with a recall of 89%, and the F1 score is 85%. Moreover, Fig. 6 shows noticeable changes in the SVM classifiers' detection performance when the time-window increases. For example, when the time-window equals 20 minutes, the detection results declined significantly to a precision of 7%, recall of 35%, and an F1 score of 11%.

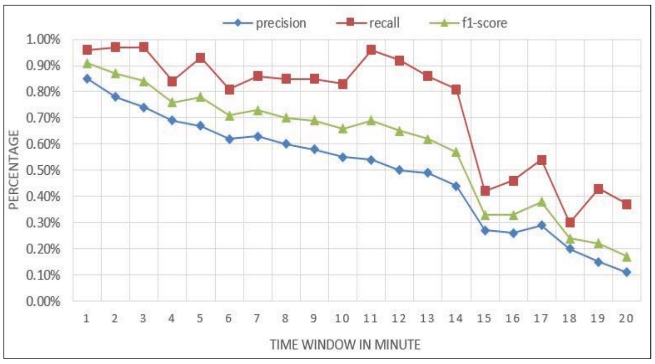

Fig. 7 NB on the Simulation Environment

Fig. 7 shows the metric values for the NB classifier models results. The NB model obtained the highest detection results when the time-window was 1 minute, with corresponding precision of 85%, recall of 96%, F1 score of 91%. The lowest precision was 11% with a 37% recall rate, F1 score of 17%. In our simulation environments, we found that the 1-minute time-window is proper for the NB classifier to detect the attack traffic. The experimental results show the critical role that the time-window of OpenFlow traffic analysis plays on the detection performance of our classifiers.

B. *Impact of Time-Window Variations*

To better quantify the impact of different time-windows of OpenFlow traffic analysis on the detection performance of the K-NN, SVM, and NB classifiers, we conducted a false-negative analysis. The purpose of a false-negative analysis is to find all the samples that are falsely identified by our classifiers and the time slot of each sample. In our application, the 'false negative' samples represent the attack samples that were falsely identified as benign samples by the trained models. We conducted the false-negative analysis after performing the physical and simulated environment experiments by allocating the attacks samples for each experiment dataset with corresponding trained K-NN, SVM, and NB models, whereas, each experiment dataset represents a different time-window of OpenFlow traffic analysis. Therefore, fed the attacking samples to the trained models and extract the samples that are falsely identified by the models as a normal sample along with the time slot of that sample. Based on our false-negative analysis results, we discovered that most of the false negatives occurred at the end of the attack time. As shown in Figures 2-7, the recall ratios, precision ratios, and F1 scores in the experiments on both environments decreased significantly when the time-window was equal to 15-minutes or longer. The reason behind the increasing number of false negatives when the time-window increased is due to the behavior of the SDN environment when it is under a saturation attack. Technically, in the early stages of a saturation attack (i.e. when the attack is initiated), both the switch and the controller have enough capacity to process the incoming attack packets. Also, the OpenFlow connection channel has sufficient bandwidth to transfer the OpenFlow messages at this time. This led to a significant increase in the numbers of Packet-In, Packet-Out, Packet-Mod, and TCP-ACK messages in the OpenFlow traffic. In this situation, the K-NN, SVM, and NB classifiers were able to accurately identify the attack samples from the benign samples, as evidenced by the high precision ratios, recall ratios, and F1 scores of the classifiers.

Subsequently, as the attack takes over the SDN network, the OpenFlow switch and the controller become overwhelmed. At this point, they do not have sufficient capacity to process the huge amount of malicious traffic, and the OpenFlow channel is also congested. Thus, the occurrences of Packet-In, Packet-Out, Packet-Mod, and TCP-ACK messages in the malicious OpenFlow traffic are similar to the occurrences of these messages in the benign OpenFlow traffic. Therefore, the K-NN, SVM, and NB classifiers are more likely to falsely identify the attack samples that are similar to benign sample as normal samples, which leads to an increase in the number of false negatives. This, in turn, reduces the recall ratios, or falsely identify the benign samples as attack samples which in turn increases the false positives and decrease the precision ratio. As a result, the overall detection performance of the machine learning classifier suffers.

C. *Detection of Unknown Attacks*

An unknown attack is an attack which has been mislabeled by the training model due to the absence of similar samples in the training dataset. In our experiments, we excluded the targeted attack and its combination of samples from the training dataset, in order to present it as an unknown attack to the trained model. In this case, the training dataset included benign traffic samples, as well as the remaining attacks and their respective samples. The testing dataset consisted of the unknown attack samples, as well as randomly selected benign traffic samples. For example, given that X is the training dataset that consists of attacks samples and normal traffic samples, Y is the testing dataset that includes attack samples and normal traffic samples, G is the unknown attack samples only, and C is the attack combination samples, the X training datasets are in the form of $X_{(\text{trainingSet})} = X - G - C$ and $Y_{(\text{testingSet})} = G + NormalTrafficSamples$.

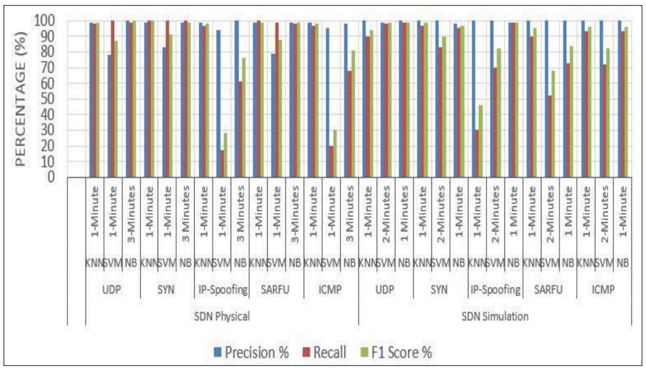

Fig. 8 Unknown Saturation Attacks Detection Results

In addition, we studied the impact of different time-windows of traffic analysis on the detection results by selecting the proper time-window of OpenFlow traffic analysis for each classifier in each environment based on the previous experiments results. Fig. 8 summarizes the detection performance results of our classifiers in both physical and simulated SDN environments. Based on the reported results, our classifiers are capable of detecting the unknown saturation attacks in both environments. In particular, the K-NN classifier shows promising detection results in both SDN environments. Also, the reported results show that the detection performance of our classifiers were influenced by the SDN environment setup. For instance, the SVM classifier obtained a 78% precision ratio, a 17% recall ratio, and a 28% F-1 score for detecting the IP-Spoofing attacks in the physical environment, whereas, it obtained 100% precision, 70% recall, and an 82% F-1 score in the simulated environment for the same attack. Based on these results, we believe that there is a relationship between the SDN environment setup and the detection performance of our classifiers.

Nonetheless, we have demonstrated that the classifiers are capable of detecting unknown saturation attacks with reasonable accuracy. We believe this is due to several characteristics of the problem. For one 1), the saturation network attacks have a high-degree of self-similarity. [20] studied the self-similarity characteristics of benign and malicious OpenFlow traffic. Their results show that the normal OpenFlow traffic has a low degree of self-similarity and has different statistical characteristics, whereas, the saturation attacks OpenFlow traffic has a higher degree of self-similarity. Secondly, (2) our features can accurately reflect abnormal behavior within OpenFlow traffic because they represent the main messages of the OpenFlow v1.3 protocol. In other words, the models are sensitive to any abnormal activity that occurs in the OpenFlow traffic between the control and data planes, because this activity is encoded in the features we have chosen for our datasets. Essentially, all the saturation attacks exhibit a similar technique of flooding the SDN environment by generating a vast number of table-miss packets; therefore, they have a similar impact on the OpenFlow messages.

VI. Conclusions

We have studied the K-NN, SVM, and NB classifiers for the detection of saturation attacks in physical and simulated SDN environments. The experiment results have demonstrated that the time-window of OpenFlow traffic has a noticeable impact on the detection performance and that the classifiers were capable of detecting unknown types of saturation attacks in SDN.

Our current work has focused on supervised classifiers in single-controller SDN environments. Our future work will apply unsupervised machine learning algorithms to the detection of known and unknown saturation attacks in SDN environments with multiple controllers.

References

[1] D. Kreutz, F. Ramos, and P. Verissimo. "Towards secure and dependable software-defined networks." In Proceedings of the second ACM SIGCOMM workshop on Hot Topics in Software Defined Networking, ACM, 2013. pp. 55-60.

[2] J. Collings and J. Liu, "An OpenFlow-based prototype of SDN-oriented stateful hardware firewalls," 2014 IEEE 22nd International Conference on Network Protocols, Raleigh, NC, 2014, pp. 525-528.

[3] C. Yoon et al., "Flow Wars: Systemizing the attack surface and defenses in software-defined networks," IEEE/ACM Transactions on Networking, vol. 25, no. 6, pp. 3514-3530, Dec. 2017.

[4] D. Kotani and Y. Okabe, "A packet-in message filtering mechanism for protection of control plane in openflow networks," in Proceedings of the 10th ACM/IEEE Symposium on Architectures for Networking and Communications Systems. ACM, 2014, pp. 29–40.

[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] A. Aizuddin, A. Ariff, M. Atan, M. Norulazmi, M. Noor, S. Akimi, and Z. Abidin. "DNS amplification attack detection and mitigation via sFlow with security-centric SDN." In Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication, p. 3. ACM, 2017.

[8] R. Sommer and V. Paxson, "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection," 2010 IEEE Symposium on Security and Privacy, Berkeley/Oakland, CA, 2010, pp. 305-316.

[9] J. Ashraf and S. Latif, "Handling intrusion and DDoS attacks in Software Defined Networks using machine learning techniques," 2014 National Software Engineering Conference, Rawalpindi, 2014, pp. 55-60.

[10] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approachfor network intrusion detection system," inProc. 9th EAI Int.Conf. Bio-Inspired Inf. Commun. Technol., 2016, pp. 21–26.

[11] D. Hu, P. Hong and Y. Chen, "FADM: DDoS Flooding Attack Detection and Mitigation System in Software-Defined Networking," GLOBECOM 2017 - 2017 IEEE Global Communications Conference, Singapore, 2017, pp. 1-7.

[12] R. Braga, E. Mota and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," IEEE Local Computer Network Conference, Denver, CO, 2010, pp. 408-415.doi: 10.1109/LCN.2010.5735752.

[13] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi and M. Ghogho, "Deep learning approach for Network Intrusion Detection in Software Defined Networking," 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM), Feb, 2016, pp. 258-263.

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

[15] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Avant-guard: Scalable and vigilant switch flow management in software-defined networks," in Proc. ACM SIGSAC Conf. Comput. Commun. Security, 2013,pp. 413–424.

[16] M. Zhang, G. Li, L. Xu, J. Bi, G. Gu, and J. Bai. "Control plane reflection attacks in SDNs: new attacks and countermeasures." In International Symposium on Research in Attacks, Intrusions, and Defenses,. Springer, Cham, 2018. pp. 161-183.

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

[18] H. Deng, Q. Zeng, and D. P. Agrawal, "SVM-based intrusion detection system for wireless ad hoc networks," 2003 IEEE 58th Vehicular Technology Conference. VTC 2003-Fall (IEEE Cat. No.03CH37484), Orlando, FL, 2003, pp. 2147-2151 Vol.3.

[19] S. M. Mousavi and M. St-Hilaire, "Early detection of DDoS attacks against SDN controllers," 2015 International Conference on Computing, Networking and Communications (ICNC), Garden Grove, CA, 2015, pp. 77-81.

[20] Z. Li, W. Xing, and D. Xu, "Detecting saturation attacks in software defined networks," in Proc. of the IEEE International Conference on Intelligence and Security Informatics(ISI). IEEE, 2018, pp. 1–6.

[21] http://mininet.org/

[22] http://www.grid.unina.it/software/ITG/

[23] https://trex-tgn.cisco.com/

[24] https://ostinato.org/

[25] https://nmap.org/nping/

[26] https://resources.infosecinstitute.com/loic-dos-attacking-tool/

[27] https://resources.infosecinstitute.com/loic-dos-attacking-tool/