Deceiving Machine Learning-Based Saturation Attack Detection Systems in SDN

Samer Y Khamaiseh
Dept. of Computer Science & Software Eng.
Monmouth University
West Long Branch, NJ, USA
skhamais@monmouth.edu

Izzat Alsmadi
Dept. of Computing and Cyber Security
Texas A&M, San Antonio
San Antonio, TX, USA
izzat.alsmadi@tamusa.edu

Abdullah Al-Alaj
Department of Computer Science
Virginia Wesleyan University
Norfolk/Virginia Beach, VA, USA
aalalaj@vwu.edu

***Abstract*—Recently, different machine learning-based detection systems are proposed to detect DDoS saturation attacks in Software-defined Networking (SDN). Meanwhile, different research studies highlight the vulnerabilities of adapting such systems in SDN. For instance, an adversary can fool the machine learning classifiers of these systems by crafting specific adversarial attack samples, preventing the detection of DoS saturation attacks. To better understand the security properties of these classifiers in adversarial settings, this paper investigates the robustness of the supervised and unsupervised machine learning classifiers against adversarial attacks. First, we propose an adversarial testing tool that can generate adversarial attacks that avoid the detection of four saturation attacks (i.e., SYN, UDP, ICMP, and TCP-SARFU), by perturbing different traffic features. Second, we propose a machine learning-based saturation attack detection system that utilizes different supervised and unsupervised machine learning classifiers as a testing platform. The experimental results demonstrate that the generated adversarial attacks can reduce the detection performance of the proposed detection system dramatically. Specifically, the detection performance of the four saturation attacks was decreased by more than 90% across several machine learning classifiers. This indicates that the proposed adversarial testing tool can effectively compromise the machine learning-based saturation attack detection systems.**

***Index Terms*—software-defined networking, adversarial attacks, DoS saturation attacks, machine learning-based detection systems**

I. Introduction

Software-Defined Networking (SDN) provides a simple way of controlling and managing networks by decoupling the control plane from the data plane. It facilitates the development and deployment of network applications by exposing programmable interfaces. However, SDN is vulnerable to DoS saturation attacks [1]. Different studies have been introduced to tackle this issue by developing machine learning-based detection systems. vSwitchGuard [2] uses the Variational Autoencoder classifier to detect and mitigate the known and unknown saturation attacks against the OpenFlow switches. SA-Detector [3] adapts self-similarity to detect DoS attacks. FDAM [4] utilizes the Support Vector Machine (SVM) classifier to detect UDP, ICMP, and IP-Spoofing attacks. Despite the promising results in using machine learning-based detection systems to detect DoS saturation attacks in SDN, their adoption in operational environments is very limited [5]. This is partly because machine learning classifiers are prone to adversarial attacks [6] [7]. An adversary can make simple perturbations or obfuscations to the inputs of the trained machine learning classifier. Thus, the classification rate of the trained model will be reduced drastically. As a result, the machine learning-based detection system will not be able to detect such adversarial attacks and render the SDN environment vulnerable to DoS saturation attacks.

Few research studies have investigated the impact of the adversarial attacks against machine learning-based detection systems in SDN. Flow-Merge [8] generates adversarial attacks that force the convolutional neural network detection method to misclassify a few attacks such as TCP-SYN and ICMP. The Hydra testing tool [9] provides a simple method to generate adversarial attacks against a group of machine learning classifiers to reduce the detection of the TCP-SYN attack in SDN. However, such research proposals generate adversarial attacks that prevent limited types of machine learning classifiers from misclassifying saturation attacks. Also, the proposed tools can only generate evasion attacks for limited types of DDoS saturation attacks in SDN. Thus, for evaluating the robustness of machine learning-based saturation attack detection systems, having a testing tool that can generate adversarial attacks for a family of DoS saturation attacks is an urgent need in SDN.

In this paper, we propose a new adversarial testing tool that is capable of generating adversarial examples (i.e., evasion attacks) for four types of saturation attacks, namely TCP-SYN, UDP, TCP-SARFU, and ICMP. The generated adversarial attacks can be used to force many supervised and unsupervised machine learning classifiers to misclassify these attacks in SDN. Also, as a testing platform, we develop an SDN saturation attack detection system. The proposed detection system utilizes different supervised and unsupervised machine learning classifiers to evaluate their robustness against adversarial attacks.

The contributions of this paper are as follows:

- To the best of our knowledge, this work is among the first to propose an adversarial testing tool that can generate adversarial attack samples and prevent the SDN machine learning detection systems from detecting a family of DoS saturation attacks.

- We present an SDN machine learning-based detection system that is capable of detecting a family of DoS saturation attacks as a testing platform.
- We evaluate the robustness of the most widely used supervised and unsupervised machine learning classifiers against a wide spectrum of adversarial attacks.

The remainder of this paper is organized as follows. Section II reviews related work. In Section III, we describe the design of the saturation attack detection system. Section IV introduces the adversarial attacks. Section V presents the adversarial testing tool. In Section VI, we explain the experimental results of the saturation attack detection system using supervised and unsupervised classifiers. Section VII introduces the experiment results of the adversarial testing tool. Finally, Section VIII concludes the paper.

II. Related Work

Recently, machine learning classifiers are used to develop security solutions in different fields, such as spam detection, malware detection, image classification, and DDoS network detection systems. However, ML-based security solutions can be invaded by adversarial attacks [10]. For example, studies have presented the potential of creating adversarial attacks against image classifications. Moosavi-Dezfooli et al. [11] present the Deepfool algorithm that can generate adversarial attacks to fool deep neural networks by perturbating some inputs to the DNN image classification. Papernot et al. [12] propose Clearhans v0.1 as an adversarial attack library that can be used to generate evasion attacks of image classifications. Also, the proposed library can be used as a benchmark to develop robust machine learning models.

Moreover, different studies show the possibility of fooling machine learning-based spam detection systems. Abid et al. [13] investigate the possibility of creating evasion attacks against android malware classifiers with different levels of adversary knowledge and capabilities. The demonstrated results show that a simple obfuscation of malicious mobile applications with minimal adversary knowledge led to a decrease in the detection rate of Random Forest and ANN classifiers by 100%. Biggio et al. [14] present a simple method based on the gradient method to generate evasion attacks against PDF files' malware detection systems, with the assumption of increasing adversary knowledge and capability. The proposed method reduced the detection performance of SVM significantly.

As an emerging network architecture, SDN provides programable interfaces that promote the development of network applications, specifically machine learning-based network detection systems [15]. Few studies have investigated the robustness of machine learning detection systems in SDN against adversarial attacks. Abusnaina et al. [8] introduce the FlowMerge method that can generate evasion attack samples against convolutional neural networks (CNN) to reduce the detection rate of TCP, UDP, and ICMP attacks. This work has many limitations: (1) the generated evasion attacks have been tested using only the CNN classifier, and (2) the generated evasion attack samples were generated by assuming that the attacker has full knowledge about the type of machine learning classifier as well as its features, which is a strong assumption. Therefore, it is unclear if the attacks generated by FlowMerge can fool other machine learning classifiers, such as supervised classifiers. Aiken and Hayward [9] propose the Hydra adversarial testing tool that can generate an evasion attack to reduce the detection rate of TCP-SYN attack by a group of machine learning classifiers. The proposed testing tool generates the evasion attack of the TCP-SYN by perturbing three features: payload size, packet rate, and bidirectional traffic. Hydra has been evaluated by using the Neptun detection system. Neptun has been developed as an SDN machine learning-based detection system to detect the TCP-SYN attack by utilizing different machine learning classifiers. Experimental results show that the evasion attack can reduce the detection accuracy of the TCP-SYN attack by 100%. However, the proposed tool can generate evasion attacks for only one saturation attack type, the TCP-SYN attack.

In this paper, we propose an adversarial testing tool that can generate evasion attacks for a group of saturation attacks (UDP, ICMP, TCP-SYN, and TCP-SARFU). It significantly reduces the prediction results for a group of supervised and unsupervised classifiers. Also, we develop a machine learning-based saturation attack detection system that can utilize different machine learning classifiers as a testing framework.

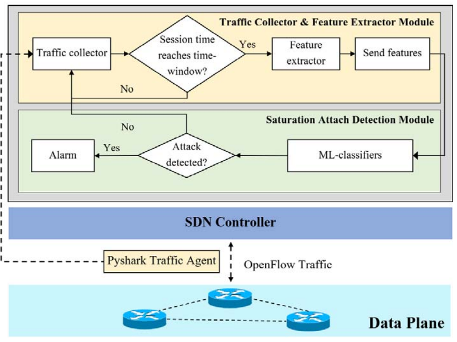


Fig. 1. The Architecture of Saturation Attacks Detection System

III. Saturation Attack Detection System

A. Design Goals

Through the design of our detection system, we aim to achieve the following two goals. First, the detection system should be effective in detecting a family of saturation attacks: TCP-SYN, UDP, TCP-SARFU, ICMP, and their combinations. Second, it should be used as a testing platform for adversarial testing tools by extending its functionality to incorporate different machine learning classifiers.

B. System Architecture

Our saturation attack detection system is designed as an application on top of the controller, as shown in Fig. 1. It consists of two modules: (1) the traffic collector and feature extractor and (2) the saturation attack detection module.

1) Traffic collector and feature extractor: The traffic collector component is a session-based process responsible for collecting the raw OpenFlow traffic promptly. Instead of using controller-to-switch flow statistics messages, which may cause performance overhead on the controller, the traffic collector employs the Pyshark library [15] to collect the OpenFlow traffic. The duration of each session is equal to a pre-defined time-window of OpenFlow traffic analysis. In this work, the time-window is equal to one minute of OpenFlow traffic analysis, as recommended in [16]. The captured OpenFlow traffic of each session is fed to the feature extractor component to extract the features to be used later by the saturation attack detection module. The feature extractor extracts four features: (1) the number of Packet-In messages forwarded from the OpenFlow switches to the SDN controller, (2) the number of Packet-Out messages sent from the SDN controller to the OpenFlow switches, (3) the number of Flow-Mod messages forwarded from the SDN controller to the OpenFlow switch, and (4) the number of TCP-ACK messages forwarded from the OpenFlow switches to the SDN controller. These features, as suggested in [16], are sensitive to saturation attacks and can reflect any abnormal behavior in the SDN environment.

2) Saturation attacks detection module: This module is a two-stage process: a training stage and a detection stage. In the training stage, before running the detection system, the selected machine learning classifier is trained using a pre-made training dataset. As a testing platform for different machine learning classifiers, we employed different supervised and unsupervised classifiers (see section III.C).

In the detection stage, the traffic collector and feature extractor modules extract the features from the live OpenFlow traffic and feed them as an instance to the trained machine learning model to determine whether the SDN environment is under a saturation attack or not.

C. Machine learning classifiers

We adopt three supervised machine learning classifiers: (1) K-Nearest Neighbors (K-NN), (3) Naïve Bayes (NB), and (3) Support Vector Machine (SVM), and two unsupervised classifiers: (1) Isolation-Forest and (2) Artificial Neural Network (ANN) for detecting saturation attacks and examining their robustness against the generated adversarial examples. We train these classifiers using a training dataset that is collected from simulated and physical SDN environments [16] and then evaluate the detection performance of each classifier using online adversarial settings.

IV. Adversarial Attacks

A. Adversary Model

The adversarial attack can be carried out against any machine learning classifier by making small perturbation to the classifier's input samples. As detailed in [6], the adversary's goals, capabilities, and knowledge of the DoS detection system should be defined in detail to mimic the real-life adversarial attack scenarios.

1) Adversary Goals: The main goal of the adversary is to launch a DoS saturation attack against the SDN network without detection by the machine learning detection system. To achieve that, an adversary can do the following: (1) reduce the trained model's confidence by increasing its prediction ambiguity, (2) increase the model's targeted-misclassifications by crafting adversarial attacks that force the trained model to produce the adversary output, and (3) increase the model's non-targeted misclassifications by applying perturbations into the domain inputs to force the output model to be any class other than the correct one. In this paper, we generate adversarial attacks that can compromise the entire SDN environment by forcing the trained machine learning classifier to misclassify the traffic of the DoS saturation attack, i.e., incorrectly classify it as a normal one.

2) Adversary capabilities: The capability of an adversary is measured by its access level to the targeted system. In some attack scenarios, an adversary has access to the classifier features and can modify the training set as well as the trained model configuration parameters [12] [7]. In other evasion scenarios, the adversary can use the classification model as an oracle. Therefore, they can conduct different queries to extract the relationships between the input and the classifier output which helps in crafting adversarial inputs. In other attack scenarios, the adversary can use the sampling technique to create evasion attacks. It can collect input-output pairs of the trained model to resolve the relationships between these inputs and outputs. However, it cannot change these inputs to observe the difference in the output. Therefore, to be able to perturb the trained model inputs, the adversary needs to collect a large number of these pairs in order to identify these relationships.

3) Adversary knowledge: The adversary's prior knowledge can be categorized into different categories, such as the knowledge about the detection measurements used by the detection system, the used machine learning classifier, the classifier output, the feature vectors, the network topology, and network architecture

B. Threat Model

In this paper, we assume that the adversary has zero knowledge about the machine learning classifier, the trained model features, or the training dataset. Also, the adversary can make assumptions about the employed classifier type and features that could invade the trained model. Furthermore, the adversary has access to a single machine (i.e., zombie host) that is connected to an OpenFlow switch. Finally, it can perturb three network traffic parameters, as explained in IV.C.

C. Adversarial Perturbation Network Features

The main strategy to craft adversarial attack samples is based on perturbing some significant features of the network traffic that have the potential to be used by the machine learning-based detection system. In this work, we defined the change rate of IPv4 source address, the packet rate, and the change rate of Ethernet source address as fundamental features to generate evasion attacks. These features have a substantial impact on the behavior of the OpenFlow traffic of the SDN environment. Primarily, the OpenFlow switch uses match-action tables (i.e., flow tables) to process the incoming packets. It uses the incoming packet header information, such as the IPv4 source address and Ethernet source address, to match it against the flow rules and process the packet based on the corresponding action. When the incoming packet does not match any of the flow-rules, a table-miss occurs. At this point, the OpenFlow switch generates a Packet-In message, which includes the table-miss packet and sends it to the controller. Consequently, the controller will decide the fate of this new incoming packet and inform the OpenFlow switch by sending Packet-Out and Flow-Mod messages.

This reactive packet processing exposes a security threat. An attacker can employ the TCP-SYN, UDP, ICMP, and TCP-SARFU flooding attacks to launch data-to-control plane attacks. This is accomplished by sending table-miss packets at a high rate by frequently changing their IPv4 source address and/or Ethernet source address. The reason behind this frequent change is to reduce the possibility of finding a matching flow entry in the targeted OpenFlow switch and thus generate a large number of Packet-In messages. As a result, the controller will be overwhelmed by processing the Packet-In messages, which will consume its computational resources. Next, a large number of Packet-Out and Flow-Mod will be forwarded by the controller, which will exhaust the memory of the OpenFlow switch, and hence the OpenFlow connection channel will be saturated. This elucidates the reason behind using the number of Packet-In messages and other OpenFlow messages as features by different machine learning detection systems [2] [15]. As critical parameters to DoS saturation attacks, these features can be used to generate samples of adversarial attacks by careful perturbation to their values as follows:

1) The change rate of IPv4 source address and Ethernet source address : The fast and frequent change in IPv4 source addresses and/or Ethernet source address of the attack packets urges the targeted OpenFlow switch to generate a large number of Packet-In messages that could overwhelm the controller. Fig. 2. illustrates the relationship between the number of IPv4 source addresses that cause table-miss packets and the amount of generated Packet-In messages. Increasing the number of new IPv4 addresses will elevate the possibility of a table-miss, which will increase the generation of the Packet-In message. Also, decreasing the number of new IPv4 addresses will decrease the number of Packet-In messages. Therefore, we can generate evasion attacks by lowering the change rate of the IPv4 source addresses and Ethernet source addresses. Thus, the number of Packet-In messages generated by the targeted OpenFlow switch will be decreased, which in turn, lowers the number of Flow-Mod, TCP-ACK, and Packet-Out messages generated by the controller. As a result, the evasion attack traffic will be similar to a high volume of normal OpenFlow traffic. Thus, it will be difficult for the detection system to classify the evasion attack traffic as malicious one, as explained in our experiments Section VII.

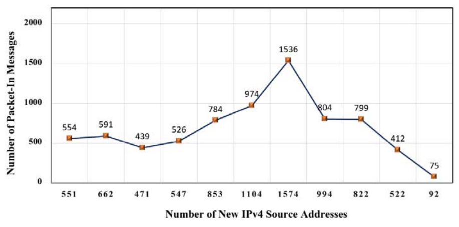


Fig. 2. The Impact of Changing IPv4 Source Address on the Packet-In Message Generation

2) Packet rate: In DoS saturation attacks, sending the attack packets at a high rate is a key to deplete the resources of the SDN environment [2]. Some SDN detection systems adopt the rate of table-miss packets as an indicator of a DoS saturation attack [17]. Sending the attack packets at a lower rate, but fast enough to overwhelm the SDN environment, will reduce the number of Packet-In messages, which will make the malicious OpenFlow traffic similar to a benign one. Therefore, the misclassification rate of the trained model will be increased, and the detection rate will be decreased, as will be described in our experiments in Section VII.

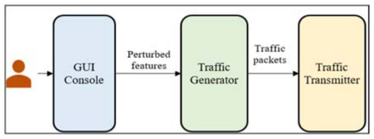


Fig. 3. The Architecture of Adversarial Attack Testing Tool

V. The Proposed Adversarial Testing Tool

A. Design Goals

The proposed testing tool should achieve two design goals, as follows: (1) It should be able to generate evasion attack examples for four types of saturation attacks (UDP, ICMP, TCP-SARFU, and TCP-SYN), and (2) It should provide a user interface to customize the perturbation of the traffic features.

B. Architecture

Fig. 3. shows the proposed adversary testing tool architecture. It consists of three modules, as follows:

1) GUI Console: The graphical user interface (GUI) provides a flexible way to select the attack features and their perturbed values. Through the GUI, the adversary can select the attack type (TCP-SYN, UDP, TCP-SARFU, and ICMP), and the IPv4 address and port number of the targeted host. Additionally, the adversary can configure the perturbations settings of the following: (1) change rate of IPv4 source address, which represents the number of different IPv4 source addresses per second used to launch evasion attack packets, (2) change rate of the Ethernet source address which represents the number of different Ethernet source addresses per second used to launch evasion attack packets, (3) and packet rate which represents the number of generated packets per second (PPS).

2) Traffic generator: This module is responsible for generating the evasion attacks traffic and then sending it to the traffic transmitter module. It supports the traffic generation of different layer-4 protocols: TCP, UDP, and ICMP. Also, it can synthesize the attack traffic packets based on the received values of perturbed features such as the IPv4 source address, Ethernet address, and the packet flags such as the SYN flag. This module can generate the evasion attack traffic packets at different rates, which are specified by the user with fixed packet size (512 bytes). The minimum packet rate is 100 PPS, and the maximum rate is 1000 PPS. Upon completing the traffic generation, the traffic packets will be forwarded to the traffic transmitter module.

3) Traffic transmitter: This module is triggered as soon as the traffic generation module completes the generation of evasion attack traffic. It is responsible for sending the traffic to the targeted host. Before transmitting the traffic, it tries to open a connection channel with the targeted host. The targeted host represents any machine connected to an OpenFlow switch. The targeted host is determined by using the IPv4 address and port number provided by the attacker. Once the connection is established, the traffic transmitter module sends the evasion attack traffic.

C. The working process of the adversarial testing tool

Initially, using any host (i.e., zombie host) connected to an OpenFlow switch, the attacker launches the adversarial testing tool and selects the attack protocol type. He can exclusively select one of TCP-SYN, UDP, ICMP, and TCP-SARFU at a time. Next, the attacker selects the perturbation values for one or more of the following features: the change rate of the IPv4 source address, the change rate of Ethernet source address, and the packet rate. Consequently, the traffic generation module will be triggered to generate evasion attack traffic based on the selected values. Then, the traffic transmitter module receives the traffic from the traffic generation module and sends it to the targeted host using the IPv4 address and port number provided by the attacker.

VI. Experiments of Saturation Attacks Detection System

A. Implementation

We implemented our detection system, including the traffic collector and feature extractor module and saturation attacks detection module. Each one of them is implemented as an application on Floodlight master v1.2 in Python.

B. Experimental setup

First, we installed the Floodlight master v1.2 controller on a computer equipped with Core-i7 CPU and 32GB RAM. To create an SDN environment, we used the Mininet network emulation tool. The created SDN simulated environments have different network topologies and network scales. The network typologies include: mesh, Star, line, and tree. The network scale is divided into three categories: (1) small scale (10-20 switches and 10-50 hosts), (2) medium scale (30 -90 switches

TABLE I
DEFAULT PERTURBATION VALUES

Feature	**Value**	**Description**
IPv4 source address change rate	10 - 50 PS	Number of different IPv4 source address per second
Ethernet source address change rate	10- 50 PS	Number of different MAC source address per second
Packet Rate	100 - 1000 PS	Number of transmitting packets per second

and 60-180 hosts), and (3) large scale (100 – 200 switches and 200-400 hosts).

Second, before running the detection system, the utilized machine learning classifier needs to be trained in advance using a pre-made training dataset. The training dataset is from our prior work [16]. It was created by using both simulated and physical SDN environments. The normal OpenFlow traffic was created using four traffic generator tools: Ostinato, Cisco Trex, D-ITG, and Nping to mimic the real-life network traffic. For the malicious traffic, Hping3 was used to generate TCP-SYN, UDP, ICMP, TCP-SARFU, IP-Spoofing, and their combinations. The total generated benign OpenFlow traffic is 393GB for a total duration of 237 hours, and the total generated malicious traffic is 150GB for a total duration of 26.2 hours. The time-window of the training dataset is equal to one-minute of OpenFlow traffic. The dataset features are the number of Packet-In messages, the number of TCP-ACK messages, the number of Flow-Mod messages, and the number of Packet-Out messages.

Third, as a testing platform, we evaluate the detection performance of K-NN, SVM, and NB as supervised classifiers and ANN and Isolation-Forest as unsupervised classifiers. Each of the employed classifiers has been evaluated independently. For each classifier, we conducted 15 experiments that cover TCP-SYN, ICMP, TCP-SARFU, and UDP saturation attacks and their combinations. We used the Hping3 tool to launch these attacks against our detection system.

Fourth, we utilized the recall and precision evaluation metrics to evaluate the detection performance of our system using different machine learning classifiers. The recall and precision are defined as:

$$Recall = TP/(TP + FN) \tag{1}$$

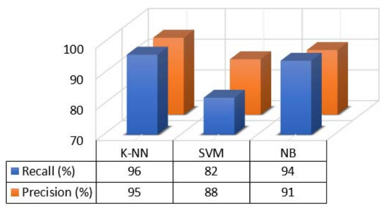
$$Precision = TP/(TP + FP) \tag{2}$$

where TP is the true positive, FN is the false negative, and FP is the false positive.

C. Experimental Results

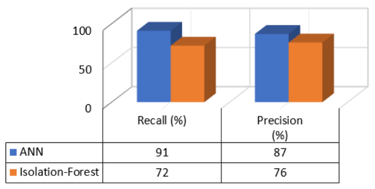
We measured the detection performance of our detection system using the supervised classifiers (K-NN, SVM, and NB) as shown in Fig.4. The proposed detection system was capable of detecting the TCP-SYN, ICMP, TCP-SARFU, and UDP saturation attacks in each experiment. Our detection system achieved the highest detection results by utilizing the K-NN classifier with 96% recall and 95% precision.

Similarly, we evaluated the detection performance of two unsupervised machine learning classifiers: the ANN and the Isolation-Forest classifiers. As depicted in Fig.5. using the ANN classifier, our detection system can detect the saturation attacks effectively with 91% recall and 87% precision. However, using the Isolation-forest classifier, the detection system produced a few false-positives that decreased its detection performance. Based on the reported results, the proposed detection system can effectively detect the UDP, TCP-SYN, TCP-SARFU, and ICMP saturation attacks using both the supervised and unsupervised classifiers.



	K-NN	SVM	NB
🟦 Recall (%)	96	82	94
🟧 Precision (%)	95	88	91

Fig. 4. The Average Detection System Performance using Supervised Classifiers.



	Recall (%)	Precision (%)
🟦 ANN	91	87
🟧 Isolation-Forest	72	76

Fig. 5. The Average Detection System Performance using Unsupervised Classifiers.

VII. Adversarial Testing Tool Experiments

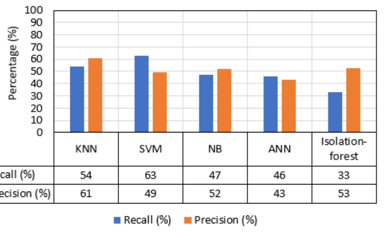
A. *Implementation and Experiment Setup*

We implemented our adversarial testing tool, including the GUI console, the traffic generator, and the traffic transmitter in Python. Next, we set up our testing environment as follows. First, we created the simulated SDN environment using the Mininet tool and Floodlight controller v1.2 as an SDN controller. Second, we deployed our saturation attack detection system into the SDN environment. Third, we deployed our testing tool into the SDN environment by installing it into a connected machine (i.e., zombie host) to launch the evasion attacks against the running detection system.

To evaluate the effectiveness of evasion attacks generated by the proposed testing tool, we performed 140 online experiments. For each classifier utilized by our detection system, we conducted 28 experiments that cover the selected features outlined in Section IV.C and their combinations per attack. The generated evasion attacks cover TCP-SYN, UDP, ICMP, and TCP-SARFU attacks. The perturbation values for each feature are depicted in TABLE I. For each experiment, we recorded the recall and the precision, as well as the perturbated feature and the corresponding values.

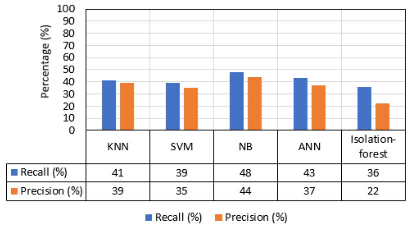
B. *Experimental Results*

To verify the impact of each feature on the detection performance of the classifiers utilized by our detection system, we perturbed one feature and kept the remaining constant. As depicted in Fig. 6, the detection performance of all classifiers declined when we reduced the change rate of the IPv4 source



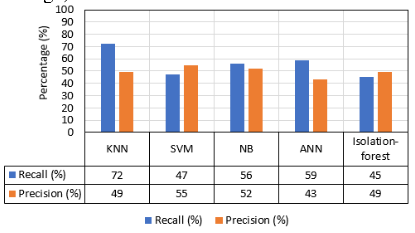
	KNN	SVM	NB	ANN	Isolation-forest
🟦 Recall (%)	54	63	47	46	33
🟧 Precision (%)	61	49	52	43	53

Fig. 6. The impact of perturbing the IPv4 Source Address Changing Rate on ML-Classifiers Detection Performance (Average).



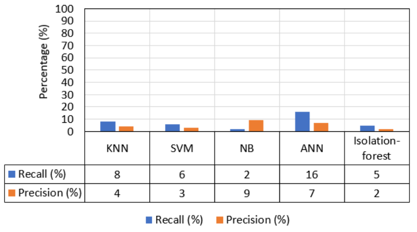
	KNN	SVM	NB	ANN	Isolation-forest
🟦 Recall (%)	41	39	48	43	36
🟧 Precision (%)	39	35	44	37	22

Fig. 7. The Impact of Perturbing Packet Rate over ML-Classifiers Detection Performance (Average).



	KNN	SVM	NB	ANN	Isolation-forest
🟦 Recall (%)	72	47	56	59	45
🟧 Precision (%)	49	55	52	43	49

Fig. 8. The Impact of Perturbing Ethernet Source Address over ML-Classifiers Performance (Average).



	KNN	SVM	NB	ANN	Isolation-forest
🟦 Recall (%)	8	6	2	16	5
🟧 Precision (%)	4	3	9	7	2

Fig. 9. The Combined Perturbation Impact on the Detection Performance of ML-Classifiers (Average).

address of the saturation attack packets. For example, with the K-NN trained model, the recall decreased from 96% to 54%, and the precision declined from 95% to 61%. This is partially related to the behavior of the SDN environment. The reduction of the IPv4 source address rate led to a decrease in the number of generated Packet-In messages. Thus, the number of Packet-Out and Flow-Mod messages was reduced. With less volume, malicious traffic started to resemble benign traffic. Thus, the classifier falsely classified the malicious OpenFlow traffic as normal, which explains the reduction of the classifiers' recall rates due to the increase in false negatives (i.e., attack traffic classified as normal). Also, it illustrates the reason behind the decline in the precision rate of these classifiers because of the significant increase of their false positives (i.e., benign traffic classified as attack traffic).

As illustrated in Fig. 7, perturbing the saturation attack packet rate can have a huge impact on the detection performance of both the supervised and unsupervised classifiers. For instance, with the highest detection performance obtained by the NB classifier, the recall is 48%, which is decreased from 94%, and the precision is 44%, which is declined from 91%. Sending the saturation attacks at a lower rate, and fast enough, can reduce the occurrence of table-miss packets, which will reduce the occurrence of Packet-In messages. Thus, the OpenFlow traffic behavior will not be dramatically changed in a way that can alarm the trained detection model. However, lowering the packet rate of the saturation attacks will increase the attack time since the generated malicious traffic will take a longer time to consume the computational resources of the SDN environment.

As depicted in Fig. 8, the machine learning classifiers' detection performance decreased when we lowered the change rate of the Ethernet source address in the saturation attack packets. This reduction in Ethernet source address change rate plays the same role as IPv4 source address change rate. This is because the flow-rules of the OpenFlow switches can use both or one of them to match the incoming packets. Therefore, lowering the Ethernet change rate can reduce the table-miss packets. Thus, the malicious OpenFlow traffic will be similar to a normal one.

Moreover, we investigated the detection performance of the adopted classifiers when the evasion attacks are generated by perturbing all features at once. We observed that the detection performance of the classifiers utilized by the detection system is decreased significantly. As demonstrated in Fig. 9. the detection performance in terms of recall and precision is reduced by all the classifiers (supervised and unsupervised). For example, the K-NN classifier recall decreased from 96% to 8% as well as the precision decreased by 91%. With the worst detection performance, recorded by the Isolation-forest, the recall decreased from 72% to 5%, and the precision decreased from 76% to 2%. This demonstrates that the proposed testing tool can effectively generate evasion attacks that can significantly reduce the detection of UDP, TCP-SYN, ICMP, and TCP-SARFU saturation attacks across a group of supervised and unsupervised classifiers in SDN.

VIII. Conclusion

In this paper, we have presented a new testing adversarial tool for evaluating the robustness of machine learning DoS detection systems in SDN. In the case study, we developed a machine-learning detection system that can utilize different supervised and unsupervised machine learning classifiers. The experiments show that the proposed testing tool can effectively generate adversarial examples for a group of saturation attacks by perturbing three traffic features. The generated evasion attacks reduced the machine learning classifiers' detection performance for UPD, TCP-SYN, TCP-SARFU, and ICMP saturation attacks by more than 90% in terms of recall and precision.

Furthermore, through experimental results, we demonstrated that the attacker can launch evasion attacks against ML-NIDS in SDN environments with minimal knowledge and capabilities about the SDN environment as well as the ML-NDIS. Thus, we suggest integrating the adversarial testing tools with the process of evaluating the ML-NIDSs before real deployment in the operational environment.

This paper studied the generation of evasion attack samples for four types of saturation attacks. Our future work will focus on examining more adversarial examples for additional types of saturation attacks as well as machine learning classifiers.

References

[1] R. Swami, M. Dave, and V. Ranga, "Software-defined networking-based ddos defense mechanisms," *ACM Computing Surveys (CSUR)*, vol. 52, no. 2, pp. 1–36, 2019.

[2] S. Khamaiseh, E. Serra, and D. Xu, "vswitchguard: Defending openflow switches against saturation attacks," in *IEEE Computer Society Signature Conference on Computers, Software and Applications (COMPSAC)*. IEEE, 2020.

[3] Z. Li, W. Xing, S. Khamaiseh, and D. Xu, "Detecting saturation attacks based on self-similarity of openflow traffic," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 607–621, 2020.

[4] 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*. IEEE, 2017.

[5] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *2010 IEEE symposium on security and privacy*. IEEE, 2010, pp. 305–316.

[6] T. N. Nguyen, "The challenges in ml-based security for sdn," in *2018 2nd Cyber Security in Networking Conference (CSNet)*, 2018, pp. 1–9.

[7] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European symposium on security and privacy (EuroS&P)*. IEEE, 2016, pp. 372–387.

[8] A. Abusnaina, A. Khormali, D. Nyang, M. Yuksel, and A. Mohaisen, "Examining the robustness of learning-based ddos detection in software defined networks," in *2019 IEEE Conference on Dependable and Secure Computing (DSC)*. IEEE, 2019, pp. 1–8.

[9] J. Aiken and S. Scott-Hayward, "Investigating adversarial attacks against network intrusion detection systems in sdns," in *2019 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 2019, pp. 1–7.

[10] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *arXiv preprint arXiv:1611.01236*, 2016.

[11] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2574–2582.

[12] N. Papernot, F. Faghri, N. Carlini, I. J. Goodfellow, R. Feinman, A. Kurakin, C. Xie, Y. Sharma, T. Brown, A. Roy, A. Matyasko, V. Behzadan, K. Hambardzumyan, Z. Zhang, Y.-L. Juang, Z. Li, R. Sheatsley, A. Garg, J. Uesato, W. Gierke, Y. Dong, D. Berthelot, P. Hendricks, J. Rauber, R. Long, and P. McDaniel, "Technical report on the cleverhans v2.1.0 adversarial examples library," *arXiv: Learning*, 2016.

[13] Z. Abaid, M. A. Kaafar, and S. Jha, "Quantifying the impact of adversarial evasion attacks on machine learning based android malware classifiers," in *2017 IEEE 16th International Symposium on Network Computing and Applications (NCA)*. IEEE, 2017, pp. 1–10.

[14] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Joint European conference on machine learning and knowledge discovery in databases*. Springer, 2013, pp. 387–402.

[15] N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, "Survey on sdn based network intrusion detection system using machine learning approaches," *Peer-to-Peer Networking and Applications*, vol. 12, no. 2, pp. 493–501, 2019.

[16] 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.

[17] R. Wang, Z. Jia, and L. Ju, "An entropy-based distributed ddos detection mechanism in software-defined networking," in *2015 IEEE Trustcom/BigDataSE/ISPA*, vol. 1. IEEE, 2015, pp. 310–317.