Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2020.DOI

InSDN: A Novel SDN Intrusion Dataset

MAHMOUD SAID ELSAYED[1], NHIEN-AN LE-KHAC[1] AND ANCA D. JURCUT[1]
[1]School of Computer Science, University College Dublin, Ireland.
(e-mail: mahmoud.abdallah@ucdconnect.ie, { an.lekhac, anca.jurcut}@ucd.ie)

Corresponding author: Nhien-An Le-Khac (e-mail: an.lekhac@ucd.ie).

ABSTRACT Software-Defined Network (SDN) has been developed to reduce network complexity through control and manage the whole network from a centralized location. Today, SDN is widely implemented in many data center's network environments. Nevertheless, emerging technology itself can lead to many vulnerabilities and threats which are still challenging for manufacturers to address it. Therefore, deploying Intrusion Detection Systems (IDSs) to monitor malicious activities is a crucial part of the network architecture. Although the centralized view of the SDN network creates new opportunities for the implementation of IDSs, the performance of these detection techniques relies on the quality of the training datasets.
Unfortunately, there are no publicly available datasets that can be used directly for anomaly detection systems applied in SDN networks. The majority of the published studies use non-compatible and outdated datasets, such as the KDD'99 dataset. This manuscript aims to generate an attack-specific SDN dataset and it is publicly available to the researchers. To the best of our knowledge, our work is one of the first solutions to produce a comprehensive SDN dataset to verify the performance of intrusion detection systems. The new dataset includes the benign and various attack categories that can occur in the different elements of the SDN platform. Further, we demonstrate the use of our proposed dataset by performing an experimental evaluation using eight popular machine-learning-based techniques for IDSs.

INDEX TERMS Dataset, Intrusion detection system (IDS), OpenFlow, SDN, Security, Threat Vectors, Machine Learning

I. INTRODUCTION

IN conventional distributed networks, the functionality of decision making processes known as control plane and, the forwarding of network traffic (data plane) are implemented within the network devices (e.g. routers or switches). The network operators configure traffic policies (e.g. routing, switching, quality of service) on each device independently.

Recently, SDN has come to prominence to solve the inherent problems of conventional distributed networks. The key benefits of SDN is making the network more flexible and easy for management by decoupling the control plane and data plane. Thus, the new paradigm can control the entire system from a centralized remote device named the controller. The benefits of SDN encourage many commercial and industrial companies to deploy SDN solutions in their network environment for several reasons, including:

- Separating the control plane from the data plane facilitates network system management. Besides, the network becomes easier for any change or update, and therefore reducing the human mistakes.
- IT administrators can implement network devices or upgrade the network infrastructure easily without any restraint to a specific vendor.
- Centralized view of the entire network allows the SDN controller to provide a global view of the whole network.
- Developers can deploy various applications in the upper layer of the SDN system to perform network services in a virtual environment [1].
- The underneath infrastructure devices do not need any programming language. As a result, the operation cost will be decreased significantly compared to the conventional network. These enormous benefits of SDN are making its market continuously growing. As a result, it achieved more than $9.5 billion at the end of 2019 [2], and this value is expected to reach $13.8 billion by 2021, as shown in Fig. 1.

Despite the numerous benefits of SDN technology, SDN is susceptible to new security threats that can be exploited by attackers to perform different malicious tasks. If the attacker successfully accesses the SDN controller, the whole system can be exposed to critical threats. Therefore, deploying IDS

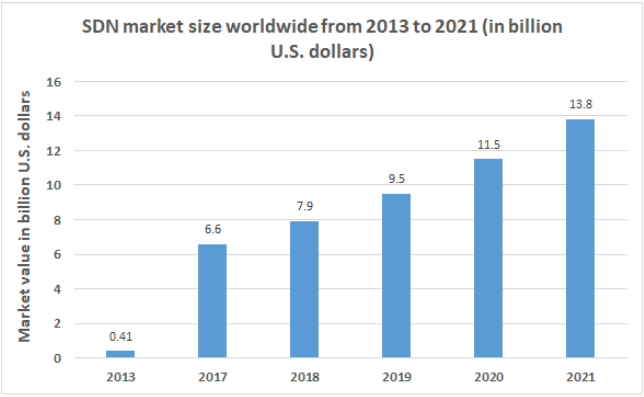

FIGURE 1: SDN market size prediction [2].

techniques to detect anomalies in the SDN network traffic is an essential part of the network architecture. Generally, IDSs can be one of two approaches: signature-based or anomaly-based solutions. While signature-based is widely used in commercial products due to its high detection rate and low false alarms, it fails to discover the new or unknown network attacks that are produced daily. In contrast, the anomaly-based detection system has gained the attention of many academic researchers due to its ability to discover novel attacks. Despite existing work conducted on the anomaly detection systems for the SDN network, unfortunately, there are still many challenges for developing efficient IDS systems on the SDN standard. One of the significant challenges for deploying IDS is the fact that there is no public dataset generated directly from SDN networks and can be used for training and evaluation of anomaly detection systems. Most of the research community uses intrusion detection datasets, which are generated for conventional networks. However, the virtualized behavior of the SDN makes the network susceptible to new attacks, which are different from those found in the conventional network.

Although some previous efforts [3]–[8] have been tried to simulate the SDN network and generate an acceptable dataset, the existing datasets only outline a few types of attacks i.e. only focus on DoS/DDoS threats without considering the different attack classes existing in the SDN network. In addition, these datasets describe intrusions that can be generated in one element of the SDN network without representing attack vectors in different SDN layers. In this work, we address the lack of available SDN datasets by generating a comprehensive dataset that contains full network traces and reflects Internet traffic. We consider the common attack classes in conventional networks, besides the new attacks data that are generated in SDN during its centralized design. The ultimate goal of this work is to create a public dataset that can be used to evaluate IDSs for the SDN environment. The contributions of this paper are summarized as follows:

- Reviewing and classifying attacks in different SDN layers.
- Studying the limitations of the existing IDS datasets.
- Proposing a virtualized network testbed to generate a new SDN dataset, namely InSDN.
- Generating a significant dataset covers various attacks that can be found in all SDN elements from the proposed network testbed. Further, the impact of the generated attacks on the different elements of SDN is reviewed. This can help the researchers to identify potential holes, and therefore, they can propose several countermeasures based on these requirements.
- Demonstrating how to use the new dataset with popular Machine Learning (ML) techniques applied in anomaly detection systems for the SDN network.

II. BACKGROUND

A. LITERATURE REVIEW

This section reviews the existing publicly datasets generated from conventional networks. These datasets are widely used for intrusion detection in conventional networks, and they have been used for evaluating ML algorithms designed for anomaly detection approaches in SDN networks.

- **KDD'99 [9] [10]:** one of the most well-known datasets which is used widely for intrusion systems evaluation. KDD'99 was derived from the DARPA packet traces. The dataset contains 41 traffic features which are classified into three groups: basic features, traffic features and content features. In addition, the dataset contains four attack categories, besides the normal data. The malicious traffic can be one of the following classes Denial of Service (DoS), Remote to Local (R2L), User to Root (U2R), or probe attacks. One of the inherent problems in the KDD'99 dataset is the redundancy records, where the duplicated records in the training set reached to 78% and about 75% in the testing file. The high degree of duplication data prevents the detection techniques to give high accuracy for low attack categories like R2L and U2R. Thus, the detection systems are biased toward frequent records like DoS attacks.
- **NSL-KDD [11]:** NSL-KDD is the modified version of KDD'99 dataset. It was produced to solve some inherent problems in the KDD'99 dataset, such as duplicate records. NSL-KDD contains two subsets, training set and testing set. The distribution of attacks in the testing set is higher than the training one, with an additional 17 attacks that are not represented in the training set. Although many studies have employed KDD'99 and NSL-KDD in the domain of intrusion detection, both datasets are not realistic to represent the current network traffic since they were generated two decades ago and cannot reflect the current attack trends. Besides, the original DARPA dataset was generated using an outdated version of the TCP protocol. Using the old TCP version makes the header field "IPv4 Type of Service (ToS)" invalid according to modern standards [12].
  Besides the previous limitations of KDD'99 and NSL-KDD datasets for IDS evaluation, they also have a large

set of features that are not relevant to SDN networks. For example, some of the previous works [13] and [14] used six out of 41 features when deploying the NSL-KDD dataset under the SDN context. The both studies selected subset features that can be derived directly through the SDN OpenFlow protocol. However, the performance of the classifier model indicates a low detection rate and a high false alarm as the used features are not being able to find the suspicious behavior of malicious traffic. In addition, we can find most of the previous works in SDN networks deployed KDD'99 and NSL-KDD datasets to identify DoS attacks only. This is because the other attack traffic like U2R and R2L are embedded in the packet's data, and the content features are required to identify these types of attacks. However, the content features are not directly accessible in the current OpenFlow protocol.

- **Kyoto dataset 2006+ [15]:** was collected from honeypot servers in Kyoto University. It contains the real network traffic in the period between (Nov. 2006 to Aug. 2009). Kyoto dataset comprises of 24 statistical features, 14 of them are shared with the KDD dataset. The background or normal traffic was created simultaneously with malicious traffic by deploying an additional server in the same honeypots network to produce a more realistic dataset. The imbalanced class distribution of the dataset is considered the main limitation of Kyoto 2006+ since the traffic data was obtained from honeypot servers, and the majority of the traffics data are malicious. Besides, the attack types in the dataset are unknown. The shortcoming to identify the attack types gives a limited view to evaluate intrusion detection performance when using this dataset. Furthermore, the normal traffic in Kyoto 2006+ covered only the mailing and DSN traces. In addition, the size of normal traffic in the dataset, i.e. between 3% and 4% of the whole dataset, does not reflect the Internet traffic. Besides, normal and malicious traffics were created in two different environments causing to the dataset being unrealistic and uncorrelated [16]. Although the Kyoto 2006+ dataset was built on real traffic data, it does not consider any information regarding the dataset attacks types. As a result, we can find difficulties in evaluating the impact of these attacks on the SDN network services.
- **ISCX2012 [17]:** The authors used two profiles to generate data traffic based on a simulated network environment. The Alpha-profiles are used to create attack traffic and Beta-profiles for normal traffic generation. The dataset includes two main types of network attacks, DoS and brute force attacks with 20 collected packet features. However, the diversity of the DoS attacks in the data is slightly small and does not cover the vulnerabilities that can be happened in different OSI layers. Furthermore, the dataset includes only HTTP traffic, which does not reflect modern traffics, where the majority of current Internet traces are based on HTTPS traffic [18]. Again, similar to KDD'99 and NSL-KDD datasets, the number of features that can be extracted from the OpenFlow protocol are not enough for machine learning eveluation.
- **CICIDS 2017 [18]:** This dataset is the closest one to our study due to it covers a comprehensive range of attack scenarios that are not addressed in the previous datasets, besides it contains the same number of gathered flow-based features. Although the CICIDS 2017 dataset is considered one of the recent datasets that attracts many researchers to develop and analyze their new models, it contains many problems and shortcomings as the following: (i) Firstly, the CICIDS 2017 dataset was released based on the foundation of ISCX2012, published in 2012. The significant difference between both datasets is the total number of extracted features. Where the CICIDS 2017 dataset contains more than 80 flow-based features compared to 20 packet features in ISCX2012. In addition, the HTTPS Beta profile was added to the CICIDS 2017 dataset to keep the adoption of HTTPS growth on the web. (ii) Secondly, normal traffic behavior was generated based on profile scripts. However, applying the concept of profiling could be problematic due to their innate complexity [19]. Furthermore, Panigrahi *et al.* (2018) highlighted some problems and shortcomings in CICIDS 2017 data [20]. The dataset has 288602 missing class labels and 203 missing information instances. In addition, the size of the CICIDS 2017 dataset is extremely huge and contains many redundant records that seem to be irreverent for any IDS training.
- **CSE-CIC-IDS2018 [21]:** The dataset is the result of a collaborative project between the Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC). Similar to CICIDS 2017 but instead, it was implemented on AWS (Amazon Web Services) computing platform. The notion of profiles is used to generate the dataset in a systematic manner. Where this dataset has two general classes of profiles, B-profiles is used to generate the normal traffic, and M-Profiles is used for attack scenarios. The dataset covers the same attack scenarios as in CICIDS 2017 dataset. However, the dataset suffers from the same inherent problems of CICIDS 2017, and also the use of synthetic traffic.

In addition to datasets described above, many data repositories have been published to cover various security domains, such as botnets [22], [23], Malware [24], [25], Port scans [26], etc. While the structure and the type of those repositories are different, we exclude them from our comparison. More details about these datasets descriptions and discussion properties can be found in [27], [28].

An important note is that although all datasets described above are normally used for IDS research on SDN network-based, these datasets were not generated from SDN plat-

forms. This would cause a compatible problem since the conventional network and SDN are different in nature. In addition, each dataset has its own requirements for security systems and benchmark datasets should be adapted to the specific environment [27]. This means that the deployment of the attack vectors should consider the new architecture. Besides, each dataset emphasizes different properties. For example, some datasets represent certain attack types such as DDoS attack, while other datasets are concerning on the label accuracy such as ISCX2012 dataset, etc.

It should also be noted that each attack has different working principles. For example, "IPsweep" and "Portscan" attacks are not considered as DDoS attacks by the conventional intrusion detection techniques [29]. However, the aforementioned attacks can be utilized to generate an extensive amount of network flows and exhaust the SDN component resources (e.g., the bandwidth of the southbound interface gets saturated). In addition, decoupling the control plane and data plane brings some new threats that are unique to SDN. Thus, selecting the improper features can lead to a significant drawback on the performance attainable by most well-known classifiers. For better illustration, Santos *et al.* [30] demonstrated that the SDN controller attacks have the worst classifications results achieved by different machine learning algorithms. This return to the fact that some of the important features used to detect the new SDN attack types are similar to normal traffic patterns due to the unique SDN architecture. For example, flow duration, which implies alive connection time (in nanoseconds), is equal to normal flow in case of SDN controller attacks. Although this attribute is widely used in the conventional networks to detect different attack types such as bandwidth attacks, this solution is inefficient to detect the SDN controller attacks.

The comparison between the public datasets and InSDN dataset is described in Table 1, while the information of attack types and their used tools are reported in Table 2.

B. REVIEWER-2: COMPARISON OF EXISTING TESTBEDS WITH PROPOSED MODEL

We created the InSDN testbed to generate a benchmark dataset for SDN. This section compares this InSDN testbed with the existing methods in the literature. The testbed is an environment designed that can incorporate real network facilities and real traffic [31]. Table 3 represents the different testbeds that are introduced in the literature to the one we developed and used to create an attack traffic dataset.

Braga *et al.* (2010) simulated the SDN network to test ML models against DDoS flooding attacks [32]. In their work, they used 16 GB RAM and Xeon server to create an SDN testbed in order to generate an SDN intrusion dataset. The simulated network composites of three virtual OpenFlow switches connected to the SDN NOX controller. The Stacheldraht tool was used to generate the DDoS flooding attacks, where the total number of extracted samples are 32425 and 39071 for DDoS and normal traffic, respectively. Although the significant amount of samples are generated for normal and attack traffic, the collected data is limited to DDoS flooding attacks.

Amaral *et al.* (2016) created a testbed with a small topology to collect network traffic data using an OpenFlow protocol for ML-based traffic classification solutions [33]. The authors used HP VAN SDN controller with a single HP E3800 OpenFlow-enabled switch to generate their dataset.

The switch is connected to the non-SDN network to receive copies of the upstream-link traffic through the mirroring port. Two different datasets were created from the testbed to represent various traffic applications such as YouTube, Vimeo, Facebook, LinkedIn, etc. The first dataset is relatively small and labeled under a controlled environment to represent eight different application traffic. The second dataset is unlabeled and contains all traffic data generated from the monitored room. However, the collected data highlights only normal application traffic without any representation for attack scenarios. The intrinsic dataset should cover normal and malicious traffic.

Ajaeiya *et al.* (2017) Used the RYU SDN controller with a single Open vSwitch (OVS) for the experimental purpose [34]. The authors used publicly available PCAP files, which were collected from different experiments for their work. TCP Replay tool was used to replicate the network traffic into the SDN network, while the Wireshark tool was used to capture the traffic samples. The authors successfully collected 16,624 and 36,654 samples for normal traffic and attack traffic, respectively. The attack samples include the Brute Force credential attack, TCP DoS, ICMP Flood, and port san traffic. However, their work is mainly focused on re-modeling traffic replay instead of addressing actual traffic generation.

In 2018, Cheng *et al.* created a testbed network topology using the Mininet tool on the Ubuntu server [5]. Five hosts are used to create network traffic. Two hosts act as bots, and two different hosts are dedicated to normal traffic, while the last host represents the victim machine. Hping3 tool is used to generate different types of flooding attacks such as ICMP flood, UDP flood, and TCP SYN food. The same tool is also used to create the normal traffic. More than 30000 samples are collected to train the ML-based models. However, their proposed work is limited for DDoS flooding attacks only.

In 2018, Prakash *et al.* used the Mininet tool to build a topology from four virtual hosts and two virtual switches in order to generate a dataset for ML classification purposes [3]. TCPDump tool is used to collect the network traffic, while the Hping3 tool is utilized to simulate DDoS attacks. In their work, 2000 and 4000 samples are collected for normal and attack traffic, respectively. However, this work focused mainly on DDoS attacks without any consideration of other attacks that can happen in the SDN network.

In 2019, Santos *et al.* created an SDN testbed to generate the attack traffic dataset for analyzing the performance of some ML techniques [30]. The SDN network was simulated using the POX controller and Mininet tool. Scapy, a packet generation tool, is utilized to generate both normal and

TABLE 1: Comparison between produced dataset and publicly available datasets according to the principal objectives.

Dataset	Year	Realistic Traffic	Label	Attack Diversity	Meta-data	Balanced	Type of Network	No. of Attributes	Format	Network Environment
KDD'99 [9]	1998	No	Yes	Yes	No	No	Small network	41	Other	conventional Network
NSL-KDD [11]	2009	No	Yes	Yes	No	No	Small network	41	Other	conventional Network
Kyoto [15]	2006 - 2009	Yes	Yes	No	No	No	honeypots	24	Other	conventional Network
ISCX2012 [17]	2012	Yes	Yes	Yes	Yes	No	Small network	Not known	Packet, Flow	conventional Network
CICIDS 2017 [18]	2017	Yes	Yes	Yes	Yes	No	Small network	83	Packet, Flow	conventional Network
CSE-CIC-IDS2018 [21]	2018	Yes	Yes	Yes	Yes	No	Small network	83	Packet, Flow	AWS platform
Proposed (InSDN)	Since 2020	Yes	Yes	Yes	Yes	No	Small network	83	Packet, Flow	SDN Network

TABLE 2: Attacks vectors between generated dataset and public datasets with the used tools if available.

DataSet	Attacks Traffic
KDD'99 [9]	DoS, Probe (Probing Attack), R2L, U2R
NSL-KDD [11]	DoS, Probe (Probing Attack), R2L, U2R
Kyoto dataset 2006+ [15]	not specified (e.g. various attacks against honeypots)
ISCX2012 [17]	HTTP DoS (executed through Slowloris), DDoS using an IRC botnet, SSH brute force (executed through brutessh), infiltration (executed through Metasploit)
CICIDS 2017 [18]	botnet (executed throughAres), DoS (executed through Hulk, GoldenEye, Slowloris, and Slowhttptest), DDoS (executed through LOIC), cross-site-scripting, SSH brute force (patator), SQL injection, heartbleed, infiltration, portscan (executed through NMap)
CSE-CIC-IDS2018 [21]	botnet (executed throughAres), DoS (executed through Hulk, GoldenEye, Slowloris, and Slowhttptest), DDoS (executed through LOIC), cross-site-scripting, SSH brute force (patator), SQL injection, heartbleed, infiltration, portscan (executed through NMap)
Proposed (InSDN)	botnet (executed through Ares), DoS attacks (executed through LOIC, slowhttptest, HULK, torshammer, Nping, Metasploit framework), DDoS (executed through Hping3), web Attacks (executed through Metasploit framework, sqlmap), Password Brute-Forcing attack (executed through Burp Suite, hydra, Metasploit framework), probe (Nmap, Metasploit framework), exploitation (executed through Metasploit framework)

malicious traffic. The network topology is composed of a single OpenFlow switch and six hosts. The same hosts are used to generate normal and malicious traffic in two different experiments. The normal traffic represents HTTP and ICMP traffic only, while the malicious traffic is limited to DDoS attacks. The data flow size is 20000 samples, with 10000 for each traffic type.

Similar to the previous work, the SDN testbed is created to simulate two types of flooding attacks: UDP flooding and SYN flooding attacks [6]. Scapy tool is used to create 2000 samples for each attack type. The Miniedit (GUI editor for Mininet emulator tool) is used on the Ubuntu server to create different virtual hosts for normal and attack scenarios. The authors utilized scapy tool also to generate both normal and attack traffic. However, their work is limited to DDoS attacks only.

In 2020, Polat *et al.* created a testbed using six virtual hosts, two virtual machines (VM) switches, and one OVS switch to generate a normal and malicious dataset for ML training purposes [35]. They used Ubuntu 18.04 server with 1 GB RAM and I CPU on VirtualBox-KVM. The generated dataset has 65000 samples for DDoS attacks and 64,000 samples for normal traffic with 12 feature attributes and one labeled class. Hping3 tool was used to generate three DDoS attack traffic packets TCP, UDP, and ICMP flooding attacks. Again, the generated dataset is limited to DDoS attacks only.

Hence, it is clear from the studies mentioned above that there have not been real attempts to generate a comprehensive dataset for the SDN environment. The current works focus only on creating a dataset that can assist researchers in deploying ML techniques to effectively analyze and detect security problems in one element of the SDN network. One of the significant shortcomings of these methods is that the simulated data is limited to one or two activities (mostly for DoS/DDoS attack) without considering various attack types that can occur in SDN networks. In addition, their works experienced several concerns, such as out of data, representation of modern attacks, data corruption, inconsistencies, and traffic verity.

The main differences between the previous testbeds and InSDN one are the attack variety and the realistic of traffic traces. We used Kali Linux to perform various attack scenarios and create different attack classes such as DoS, DDoS, Web attacks, Password-Guessing, Botnet, Exploitation, and Probe attacks. Furthermore, the normal traffic covers various popular application services that were not represented in the previous works except [33].

C. ATTACK VECTORS IN SDN ELEMENTS

The centralized design of SDN architecture introduces new vulnerabilities that can make the SDN network vulnerable to various types of security threats [36]–[38]. In fact, all SDN layers are unavoidably susceptible to different types of attacks. Some of these attacks are specific for the SDN i.e. as a result of separating the data and control plane functionality. These attacks can occur in the SDN controller or on the communication channels between the control and data plane

TABLE 3: Comparison of existing testbeds and their characteristics

Testbeds	Traffic Generator	Normal Traffic	Attack Traffic	Attack Variety	Environment
Braga *et al.* [32]	Stacheldraht tool	Yes	Yes	No	Virtualised
Amaral *et al.* [33]	Not stated	Yes	No	No	Physical
Ajaeiya *et al.* [34]	TCP Replay	Yes	Yes	Limited	Virtualised
Cheng *et al.* [5]	Hping3 tool	Yes	Yes	No	Virtualised
Prakash *et al.* [3]	Hping3 tool	Yes	Yes	No	Virtualised
Santos *et al.* [30]	Scapy tool	Yes	Yes	No	Virtualised
Myint *et al.* [6]	Scapy tool	Yes	Yes	No	Virtualised
Polat *et al.* [35]	Hping3 tool	Yes	Yes	No	Virtualised
InSDN Testbed	Various tools	Yes	Yes	Yes	Virtualised

devices. Beside, there are various attacks that are common between SDN standard and the conventional networks i.e. the attacks on the application layer or data plane elements. While some attacks are frequent and have a mild or moderate impact against the conventional networks, the impact of these attacks is escalated in the SDN. For example, in the case that the attacker successfully gets unauthorized access to a vulnerable machine or application in a conventional network, a single machine or a small portion of this network is affected by this attack. The attacker needs to escalate his privilege or uses the victim machine to start new attacks against different machines or the subnets [39]. Therefore, there is a need for different mitigation techniques to deal with them. Kreutz *et al.* (2013) defined various attack vectors that can tamper the SDN architecture [40]. [41]–[46] consider the security issues in OpenFlow. In this section, we outline the comprehensive attack vectors that have a critical impact on different elements of SDN. Figure 2 summarizes the attack vectors inside the SDN network. We classify the main attacks against the SDN network into four-vectors as following:

1) **Attacks on the data plane.** The network elements itself can be a target of the intruder. The attacker can gain unauthorized access to vulnerable hosts in the SDN network to initiate different attacks. Besides, the attacker can generate malicious traffic, using a hosting machine or connected switch to flood the network components. The main goal of these attacks is to consume the controller resources or flow table-space of any OpenFlow switch. In addition, the attacker can cause damages in the network resources by deploying a fake switch in the SDN network in order to deviate the network traffic or for stealing purposes. The intruder can manipulate the flow entries rules of OpenFlow switch to reroute the legitimate network traffic. Furthermore, he can use the fraud switch to produce forget-requests to overwhelm the controller or to slow down the network traffic. Additionally, the virtualized view of the SDN network encourages the enterprise administrators to implement software switches such as OVS switch on their network infrastructure. Although virtual switches are software-based and run on the host servers, they can also be a target for attackers. In contrast, it is significantly difficult to physically compromise the hardware switches in the conventional network and modify its forwarding tables.

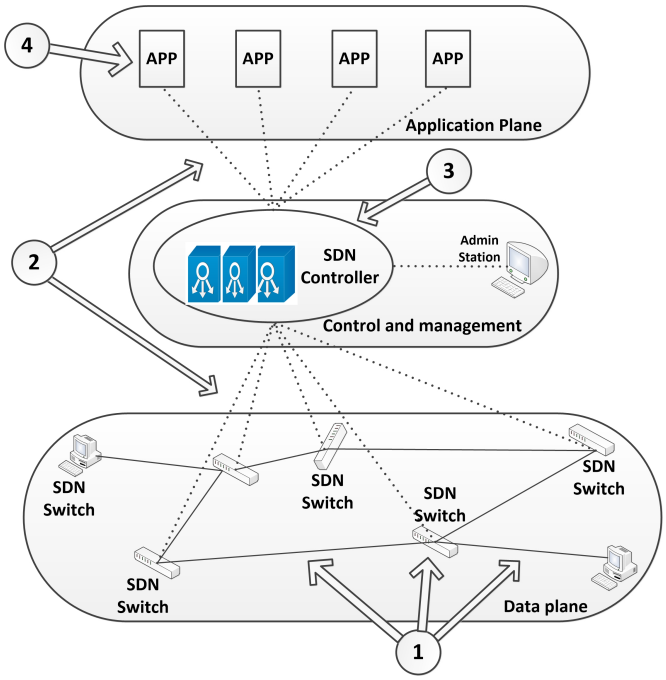

FIGURE 2: The attack vectors in different layers of SDN. We classify the SDN threats into four attack vectors. The attack vectors number 1 and 4 are shared with the conventional network, while attack vectors number 2 and 3 are particular for SDN paradigm.

2) **Attacks on Control plane Communication.** In the SDN network, the controller can handle the data plane devices through communication channels. Logically, each device has a separate channel with the controller, but physically, all these channels share the same physical link. Running the flooding attack from spoofed sources can cause congestion in the channel links. Consequently, breaking down the communication between the controller and data plane elements can isolate the SDN controller from the whole network elements.

Furthermore, the attacker can exploit the trust between the OpenFlow switches and the controller to launch a man-in-the-middle attack, sniff valuable information, or gain full access to the controller plane [47].

3) **Attacks on SDN Controller.** The controller acts as the brain of the SDN network. Gaining access or bringing down the SDN controller can consequently disrupt the whole system. In addition, the controller is vulnerable to the same vulnerabilities as the operating system installed on it. In some cases, the attacker can use his own controller and forward the node traffics based on his setup. Furthermore, the attacker can control the whole network and create his own policy if he successfully exploits the vulnerable Northbound API (i.e., the API resides in between the controller and the application layer).
4) **Attacks on the application plane.** The attacker can run a malicious application to violate the security policy or to bypass firewall and IDS Apps.

It is noticed that the attacks numbers 2 and 3 are specific to SDNs, resulting from decoupling the data and control plane, while the attacks 1, and 4 are common in both SDN and conventional network.

D. ATTACK PHASES

The main objective of attackers is to control the network system by gaining unauthorized access to network resources. He can steal vital information or disturb the network operation, causing damage in the entire system. There are five attacking steps that can be performed by malicious intruders, as follows:

1) **Reconnaissance:** The first step for the attacker before initiating his attack. In this phase, the attacker can gather some information about the target system, such as IP addresses, operating system versions, running applications, etc.
2) **Scanning:** The attacker uses the collected information from the reconnaissance phase to discover the system vulnerabilities. Consequently, he can perform different attack scenarios against the target system.
3) **Gaining Access:** In this phase, the attacker can exploit the existing vulnerabilities to gain system control. There are several methods to access the target system (eg., buffer overflow, password cracking, and session hijacking). Once the attacker successfully obtains access to the target system, he can raise his privilege to gain full access to the victim machine.
4) **Maintaining Access:** The attacker keeps his system access by installing remote shell connections using Trojans, Backdoors, Rootkits, etc. He can employ the compromised system for different purposes, such as stealing vital information or starting a new attack against different systems.
5) **Clearing Tracks:** After gaining access to the target machine, the attacker can work to hide any malicious activities in order to avoid the detection (eg., deleting the system log).

In this research work, the aforementioned attacking steps were carefully examined to generate a more realistic dataset for IDS. In addition, we have also studied the previous work in [17], [18], [48]–[50] to generate a comprehensive dataset and to take into consideration the setting up of the new environment and the different attack methodologies inside the SDN network.

III. PROPOSED SDN ARCHITECTURE

As mentioned in Section II, the attacker can exploit the vulnerable elements of the SDN network and launch several attacks such as scanning, spoofing, DoS, etc. To generate a significant dataset, we need to deploy various applications services in the testbed environment. The produced dataset should reflect the nowadays Internet attacks that can be launched in the current SDN networks. Additionally, the attack scenarios must cover the current attack vectors in different SDN elements. Furthermore, several attack scenarios are considered from different sources coming from both outside and inside the SDN network. Besides, the normal traffic in the generated data includes various popular application services such as HTTPS, HTTP, DNS, Email, FTP, SSH. We represent our topology by creating four virtual machines (VMs) using VMware Workstation on Windows 10. The first virtual machine is Kali Linux and represents the attacker server. The secondary machine is Ubuntu 16.4 and acts on the ONOS (Open Network Operating System) controller. The third is an Ubuntu 16.4 machine to serve a Mininet and OVS switch. The forth virtual machine is a Linux based on Metasploitable 2 to provide vulnerable services for demonstrating common vulnerabilities. Fig. 3 shows the testbed network architecture for the proposed solution.

The architecture of the proposed solution composes of a single OVS switch, including three OVS bridges. One of the OVS Bridges (br1) is connected to the attacker VM machine (Kali Linux). The second OVS Bridge (br2) is connected to the vulnerable Linux machine (Metasploitable2 Server). The last bridge (S1) is attached to Mininet virtual hosts. The open-source tool ONOS [51] is used to represent the SDN controller. The ONOS software is installed in a separate VM. The communication between all virtual hosts is done by L3 switching connectivity. Additionally, four virtual hosts are created using the Mininet network emulator [52], [53] to generate legitimate and malicious network traffics. Mininet is widely used by researchers to create a realistic virtual network with virtual switches, hosts, and links on a single Linux kernel virtually. Further, we used the Damn Vulnerable Web Application (DVWA) software to represent a PHP/MySQL webserver for a better description of different attacks inside the SDN network. The DVWA is independently installed from the operating system, using docker containers in the same OVS host.

The OVS switch is configured to function as L3 switching by combining the OVS software with Linux kernel routing.

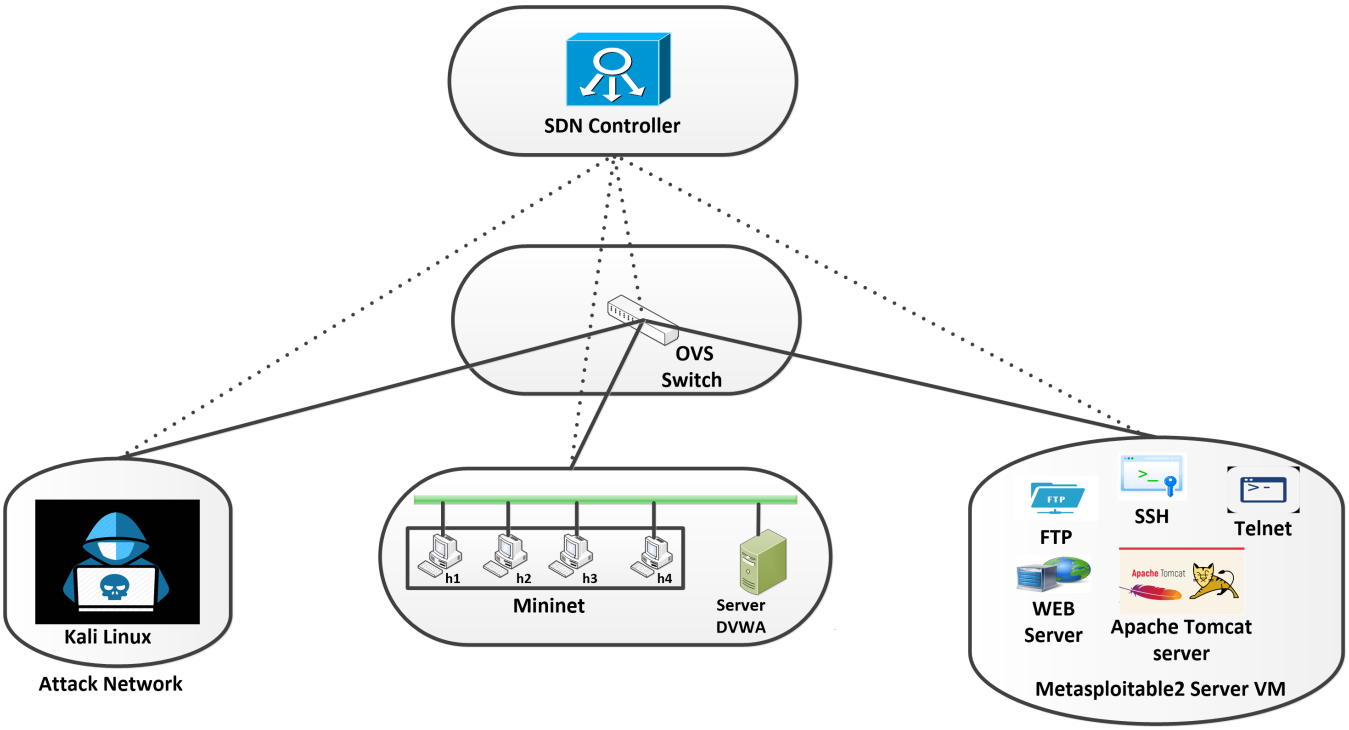

FIGURE 3: Virtual SDN testbed network architecture. The Virtual topology was created using four separated VM machines. The OVS switch and SDM controller were installed on two different machine, while the Kali Linux and Metasploitable 2 are representing the attacker machine and the vulnerable Linux server respectively.

In this case, all the virtual hosts can communicate with each other using different subnets. Fig. 4 shows the logical topology of the virtual testbed and its configuration in Fig. 5. The following process indicates how to map from L2 switching to L3 switching using the OVS switch.

- Install OVS switch and Mininet software on the same VM.
- Create four adapters in the OVS-VM to represent four different network subnets. In our setup, the created interfaces were named ens38, ens39, ens40, and ens41.
- Create two OVS bridges on the same OpenFlow switch named br1 and br2.
- Assign each data plane interface to its proper bridge. We assigned ens40 to br1 and ens38 to br2 bridge. In addition, we assigned ens41 interface into S1 bridge, which is created by default on OVS switch.
- Remove the IP address from each data plane interface or assign it to zero. Later the removed IP address will be assigned to the created bridges. For example, we remove the configured IP address from ens40 interface and assign it to its connected bridge (br1). The same configuration is performed for br2 and S1 bridges.
- Connect the Kali Linux VM with the same adapter of br1, and Metasploitable2 Server with the same adapter of br2.
- Enable IP forwarding on the OVS Linux machine.
- Create a Mininet topology that contains four virtual hosts (h1 to h4). The virtual hosts of Mininet are attached to S1 Bridge. The configuration of S1 bridge is similar to previous setups of br1 and br2. We add the IP address of S1 bridge as a default gateway for each virtual host in Mininet topology.
- Connect ONOS controller to all created bridges (br1, br2, and S1)
- Now, we are able to ping between all hosts in different subnets.

IV. METHODOLOGY FOR DATA GENERATION

A. DATASET ATTACK SCENARIOS

This section presents our approach to generate the SDN network traffic data by using different attack scenarios.

The centralized view of the SDN network and separation of the data plane from the control plane creates a new opportunity for the attacker to carry out various types of attacks compared to the conventional network. The nature of these attacks in SDN is different from those commonly affecting the conventional network [54]. For example, the attacker can generate new malicious traffic to attack the SDN controller or even the communication links between the SDN controller and OpenFlow switches. Furthermore, compromised users can be employed to start a new attacks after the traffic flow is established. Besides, the SDN applications can have different vulnerabilities such as buffer overflow, command injection, SQL injection, etc. These vulnerabilities can create attack op-

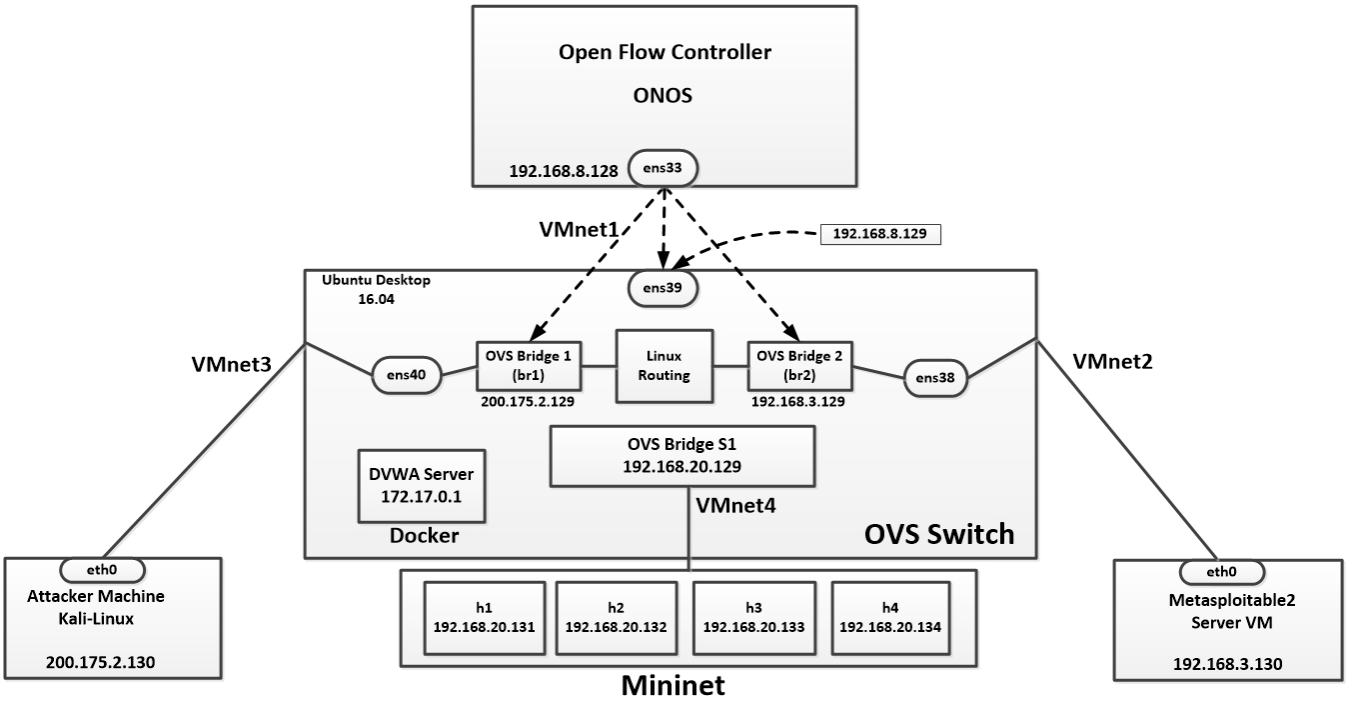

FIGURE 4: Logical Network topology. The independent OVS bridges are configured using L3 switching and connected to SDN controller

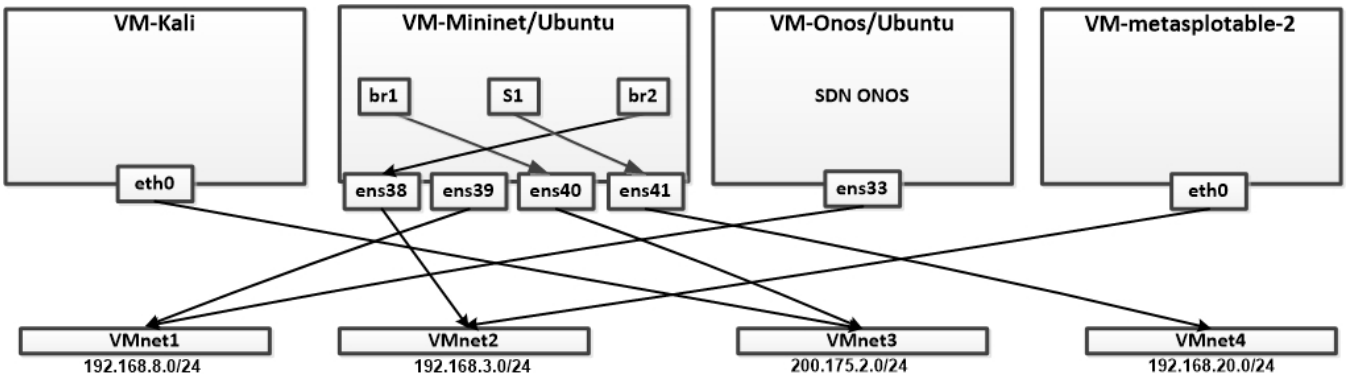

FIGURE 5: The testbed network configuration. We created four adapters to represent four different network subnets. All hosts can communicate through integrating OVS bridges with Linux kernel routing.

portunities, and help the attacker to bypass the authentication mechanism, gain access to the controller through installing a malicious script. If the attacker successfully gains access to the controller, he can start new attacks such as flow rules manipulation, launching DoS attack, and eavesdropping on the data/control traffic.

Table 4 represents the attack classes and the used tools in this virtual environment, as well as the source attack machine and the victim device IPs.

1) **DoS attacks**: Is one of the most common attacks inside the SDN architecture. It does not only damage the victim machine but can also overwhelm the SDN controller resource in a short time. Besides, the SDN controller is the brain of the SDN network, and in the case of DoS attacks, the whole system becomes unavailable for legitimate users. It turns the entire network into a 'body with no brain'. DoS attack can flood the victim machine with a huge amount of spoofed packets that have no matched rules inside flow tables switches. Thus, the OpenFlow switch will send these flows to the SDN controller in the form of `packet-In` message for further processing. When `packet-In` message rates are increased up to a certain limit, SDN controller resources can be overwhelmed by a large number of unprocessed packets. There are two main types of DoS attacks [55] as the following:

- **Network DoS attacks:** The main objective of these attacks is to overwhelm the benign users by flooding the network bandwidth or victim machine by a large amount of spoofed packets. The attacker often uses different protocols like UDP, TCP, or ICMP. DoS attacks can also disturb the SDN controller or its channels due to the significant number of forwarded packets to the controller.
- **Application DoS attacks:** Despite the fact that these attacks do not require high bandwidth, however, it can cause serious damage to the target server and consume its resources in a short time. It mainly targets the top application layer or services such as HTTP. The application layer attack is not easy to detect since the intruder is connected to the victim server in an authorized manner.

The InSDN dataset includes several types of DoS attacks that can be driven in different OSI model layers. Kali Linux is used to carry out various DoS attacks against a victim web server, which is represented by h4 virtual host. Several DoS attacks such as TCP, UDP, and HTTP flood attacks are executed by using Low Orbit Ion Canon (LOIC) tool.
Further, we implemented different slow rate DoS application attacks such as Slowloris, slow-rate HTTP POST, slowhttptest, using HULK, and torshammer tools. In addition, we handled the TCP and Slowloris based DoS attacks using the Metasploit framework on Kali Linux against Metasploitable 2 server.

2) **DDoS attacks:** InSDN dataset also includes several DDoS attacks scenarios such as TCP-SYN Flood, UDP Flood, and ICMP Flood attacks. The Hping3 tool, which considered one of the most publicly tools is used for DDoS attacks, where the attacker machines are h1 and h2, and the victim machines are h4 web server and Metasploitable 2 server.
3) **Password-Guessing Attacks:** It implies to obtain access to the victim machine through breaking the username and password credentials. Two different scenarios of Password-Guessing Attacks are considered in the InSDN dataset. In the first scenario, the dictionary attack is involved by creating a dictionary for all possible users and passwords and then try each of them. The attacker machine is Kali Linux, and the victim server is the DVWA web server. Burp suite and Hydra tools are used to launch this attack to get the username and password credentials. In the second scenario, we use auxiliary scanner tool from the Metasploit framework to discover the valid credentials on the Apache Tomcat Web server, which runs on Metasploitable 2 server.
4) **Web application attacks:** Based on the Symantec report in 2018 [56], one in ten analyzed URLs ware vulnerable with malicious code, with a 56% increase compared to 2017. In web application attacks, we implemented the most frequent application attacks such as Cross-site scripting (XSS) attack and SQL injection.
   - **XSS attack:** The attacker can bypass the access controls of the client machine by injecting malicious code into the trusted website. Once the client access the web application site, the malicious script will be executed. As a result, the attacker can obtain sensitive information from the client machine, such as session tokens, cookies, and so on. We tried to Gain Shell Access by preparing our malicious PHP file and uploaded it to the vulnerable web server. The skillful msfvenom tool, which combines Msfpayload and Msfencode tools into one single framework is used to create the PHP codes. Once the client starts to access the vulnerable web server, the uploaded PHP file will be executed. As a consequence, the attacker machine can access the infected client using a reverse connection.
   - **SQL injection attack:** The attacker can use malicious quarries to manipulate the database behind the web application, allowing the attacker to get the content of the entire SQL database. The attacker can obtain unauthorized access to any web application or sensitive data on the website. The SQL attack in InSDN dataset is executed using an automatic SQL injection (sqlmap) tool against the DVWA web server. The Burp Suite tool is used to capture the user cookies, which are needed during the SQL injection attack.
5) **Probe attacks:** This is the most essential phase for an attacker before starting his attack. The attacker scans the target system to discover some information that can assist him in exploiting the remote system such as the operating system versions, open ports, etc. We use open-source Nmap tools with different flags to run the probing attack, using Kali Linux against all virtual Mininet hosts (h1, h2, h3, and h4). Furthermore, the Metasploit framework is employed to find the open ports and the variabilities of web applications in the Metsaploitable 2 server.
6) **Botnet attack:** Although many devices or things access the Internet, the provided security does not guarantee the optimum functioning to prevent infiltration attacks. The intruder can control several infected devices, referred to botnet to run different malicious activities such as stealing information, fraud attack, launching DDoS against victim server, or web applications server. The Botnet attack is performed in the InSDN dataset by using the Ares tool, where the attacker is from the Kali Linux machine and the two hosts (h1 and h2) represent the infected bots.
7) **U2R (Exploitation) attack:** The remote exploitation and backdoor attacks are considered to represent the U2R scenario in the InSDN dataset. These malicious activities are more similar to normal traffic and can

cause a serious risk on the network system, so it is essential to detect these attacks earlier as possible [57]. In the produced dataset, we consider four vulnerable services that are running on Metsapoltable 2 server i.e., Vsftpd, distcc, UnrealRCD, and samba applications. These services are operating on corresponding ports 21, 3632, 6667, and 445, respectively. The Metasploit framework from the Kali Linux machine is used to get root access on the victim machine.

8) **Normal traffic:** We consider real Internet traffic using different protocols such as HTTPS, HTTP, SSH, mail, DNS, etc. To generate more intrinsic traffic, h3 host is connected to the Internet and run different applications like YouTube, Facebook, Email, Skype voice. Besides, several services on Metsploitable2 server are accessed to generate various samples for normal traffic such as SSH, FTP, Telnet, etc.

B. SDN SPECIFIC ATTACKS

This section analyzes several examples of attacks that can be launched in SDN elements and disrupt their normal services. We also demonstrate how these attacks can impact the SDN network severely and easily consume its resources. While some of these attacks are common with conventional networks, other attacks are more specific to SDN.

Although the SDN can be afflicted with similar attacks presented in the conventional network, the solutions that are generally applied to the current environments are not applicable for SDNs [58]. Decoupling the control plane from the data plane can bring new security threats that have never appeared in the conventional network i.e. all the unmatched packets in OpenFlow switches are forwarded to the controller in the form of `Packet-In` message. Thus, it is very easy for an intruder or even the end-user to poison the network by generating forget messages, which are relayed to the controller. If the SDN switch does not find any matching rule for the received packets in its flow tables, the switch will extract the packet header and encapsulates it using OpenFlow protocol and sent to the controller in the format of `Packet-In` message. Then, the controller encapsulates the processed flow and returns it to the OpenFlow switch in the format of `Packet-Out` message. The parameters in `Packet-Out` message, as shown in Fig. 6 are used to install the flow entry in the OpenFlow switch. The attacker can employ huge amounts of malicious requests, which will exhaust the system resources resulting in a degradation of controller performance or increasing the communication overhead.

Although the Transport Layer Security (TLS) protocol has been considered as optional to secure the communication links between the SDN controller and switches, TLS cannot protect the network from the spoofing packets.

This manuscript does not emphasize all attack types in the SDN context; instead, some attacks, which are relevant to the InSDN dataset are reported in the following paragraphs. Rather, interested readers may refer to previous studies [54], [59]–[62] for more detailed information.

- **Data-to-control plane saturation attack [63], [64]:** Different from the conventional network, unmatched packets in flow tables are forwarded to the control plane for forwarding decisions. Since the SDN controller implements the packet forwarding decisions, the attacker can exploit this vulnerability by launching a dedicated denial of service attacks to flood the network resources. He can produce an extensive amount of table-miss `Packet-In` messages to exhaust the controller's resources (eg., CPU, memory) in a short time. This can cause a reduction or complete shutdown of the controller service. As a result, the normal delivery of packets will be interrupted.
- **Link Flooding Attack (LFA) [65], [66]:** The strategy of LFA attack in the SDN context is different from those commonly targeting conventional networks. The goal of this attack is to disconnect the controller from the data plane elements. A skilled adversary can take the chance of continuous communication between the data and the control plane to obstruct this communication. For example, the attacker can generate normal packets with low rate traffic by employing malicious bots to congest the channel links by anomalous traffic, and this can impede the legitimate traffic towards the target network. However, the conventional techniques fail to mitigate it due to the centralized strategy of the SDN architecture in managing the network traffic. Besides, LFA mimics the same normal behavior during its low rate nature and can flood the whole network, without any further detection [65].
- **Flow-Rule Flooding Attack:** The attacker can flood the OpenFlow switch by creating a large amount of unmatched flow, which triggers the switch to install invalid flow rules in its entry tables. After a while, the flow tables capacity becomes full, and the OpenFlow switch is not able to install the new rules. This can deplete the switch resources and cause exhaustion in the data plane. Besides, normal users could not be able to install their flow traffic, and legal traffic cannot be forwarded.
- **Password-Guessing Attacks [43]:** An attacker residing on a non-SDN element can use random or systematic guessing of passwords to achieve unauthorized access to SDN elements. For example, an intruder might be successful in accessing a management console to launch attacks on the network managed by the SDN controller or in the controller itself.
- **Remote application exploitation [54]:** The attacker can achieve unauthorized access to a victim system or an SDN component by exploiting a software vulnerability in one of SDN components. For example, he can exploit software vulnerabilities in the application server and gain its access. If the attacker succeed to achieve unauthorized access to the application server, he can poison a controller's view of the network topology. Furthermore, the attacker can carry out a variety of other attacks such

TABLE 4: Dataset Attack Classes Generated in Virtual Environment.

Attack Classes	Description of Activities	Attack Tools	Attacker Machine	Victim-Network
DoS	TCP-ACK flood , UDP flood, HTTP flood, slow-rate, HTTP POST, Slowloris	LOIC,slowhttptest, HULK, torshammer	Kali-Linux: 200.175.2.130	Vhost(h4): 192.168.20.134
	Slowloris, TCP flood	Nping, Metasploit framework	Kali-Linux: 200.175.2.130	Metasploitable 2: 192.168.3.130
DDoS	TCP-SYN Flood, UDP Flood, ICMP Flood	Hping3	h1:192.168.20.131 h2:192.168.20.132	h4: 192.168.20.134 Metasploitable 2: 192.168.3.130
Web Attacks	XSS, Sql Inject	Metasploit framework, sqlmap	Kali-Linux: 200.175.2.130	Web server (DVWA): 172.17.0.1
R2L	Password-Guessing Attack	Burp Suite, hydra	Kali-Linux: 200.175.2.130	Web server (DVWA): 172.17.0.1
		Metasploit framework	Kali-Linux: 200.175.2.130	Metasploitable 2: 192.168.3.130
Malware	Botnet attack	ARES	Kali-Linux 200.175.2.130	h1:192.168.20.131 h2:192.168.20.132
Probe	version scan, Port Scan, discover services	Nmap	Kali-Linux: 200.175.2.130	h1:192.168.20.131 h2:192.168.20.132 h3:192.168.20.133 h4:192.168.20.134
	Port Scan, vulnerability scan (WMAP)	Metasploit framework	Kali-Linux: 200.175.2.130	Metasploitable 2: 192.168.3.130
U2R (Exploitation)	Vsftpd,IRCd, Samba and distcc	Metasploit framework	Kali-Linux: 200.175.2.130	Metasploitable 2: 192.168.3.130

as destruction of information, compromise of integrity, deviate network traffic, exploitation, and unauthorized disclosure.

V. USAGE AND AVAILABILITY

A. DATASET DESCRIPTION

We divided the dataset into three groups based on the traffic types and the target machines. The first group includes normal traffic only. The second group contains the attack traffics that target Mealsplotable-2 server. In the last group, attacks on the OVS machine are considered. The Tcpdump tool is used to capture the traffic traces for each category at the target machine and the SDN controller interface. In addition, the CICFlowMeter tool [67] is used to extract the flow features for the InSDN dataset. The reason we decided to use the CICFlowMeter in our work despite many available tools in literature such as Argus [1] and Bro-IDS [2] is the fact that none of these tools exclusively consider the time-based features [68]. However, different applications have different time constraints. As a result, it is more important to calculate the statistical time-related features for the flow traffics.

The CICFlowMeter was generated by the Canadian Institute of Cybersecurity team and has been written in Java to create network flow traffics from the PCAP file. The generated flows are calculated in Bidirectional, where the first packet in the flow determines the flow direction (forward or backward). The output of the CICFlowMeter is more than 80 statistical features in `CSV` file format such as Protocol, Duration, Number of bytes, Number of packets, etc. The list of extracted features and details are available in the appendix (Table 13). We collected more than 80 features with 56 categories from our experiments. For simplicity, we divided the entire features into eight groups as the following:

- **Network identifiers attributes:** these features contain the common information that used to define the source and destination flow. For example, IP address, Port number, protocol type.
- **Packet-based attributes:** these features hold the information related to the packets such as the total number of packets in a forward and backward direction.
- **Bytes-based attributes:** these features hold the information related to the bytes i.e. total number bytes in the forward and backward direction.
- **Interarrival time attributes:** these features show the information related to the interarrival time in both forward and backward directions.
- **Flow timers attributes:** these features hold the information related to the time of each flow i.e. active and inactive.
- **Flag attributes:** these features hold the information related to the flags like SYN Flag, RST Flag, Push flag, etc.

[1] http://qosient.com/argus/index.shtml.
[2] https://www.bro.org/index.html.

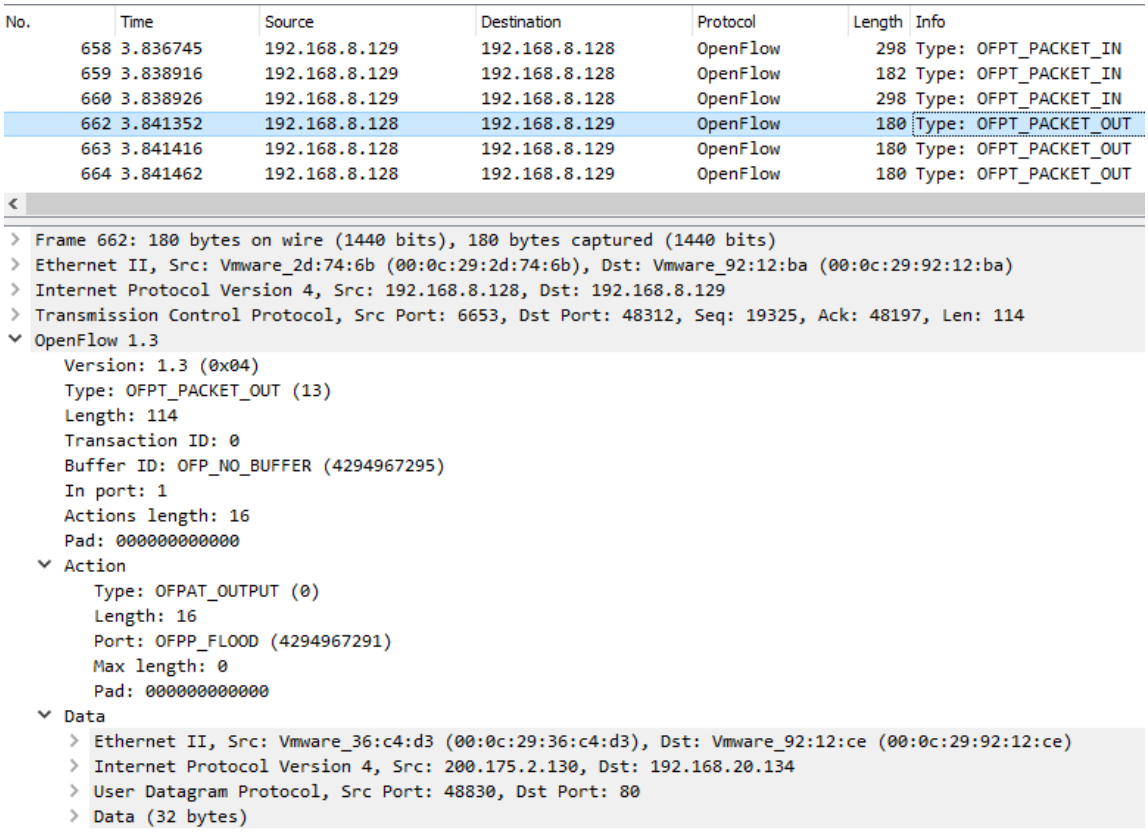

FIGURE 6: Wireshark: OpenFlow packet encapsulation.

- **Flow descriptors attributes:** these features contain the traffic flow information (eg., the number of packets and bytes in both forward and backward direction).
- **Subflow descriptors attributes:** these features show the information related to subflows, such as the number of packet and bytes in forwarding and backward directions.

For labeling processing, we use some features information such as Source IP and Destination IP. The total number of dataset instances are 343,939 for normal and attack traffic. Where the normal data brings a total of 68424, and attack traffic contains 275,515 instances. Table 5 represents the attack classes for each group with its total size. Furthermore, the name of PCAP files under each attack group is chosen based on the target protocol layer or the tools that are used to create each file.

B. USAGE NOTES

1) The InSDN dataset includes different attacks that can strike the data, control, and application layers.
   The source of attacks in the dataset is classified into two categories.
   a) **Internal:** These attacks come from internal users, who have full access to the SDN network. Although internal attacks are rare in the production systems, these attacks become more severe and can cause malicious actions for network elements. In many cases, the attacker is not able to target network servers directly since these servers might have a high level of security protection. In this case, the attacker tries to exploit weaknesses on the individual users inside the network system, and then start new attacks on different target servers. In the InSDN dataset, the compromised hosts (i.e. h1 and h2) are used to launch various attacks from internal SDN network.
   b) **External:** These attacks commonly are launched from the outside network. The attacker is mainly altering the SDN network using different malicious activities such as code exploits, DoS, malware, etc. We assume the majority of attacks in the dataset are created from an outside network to mimic the real attack scenarios.

2) We predict the effect of dataset attacks on various SDN elements. Thus, it can help to provide a better countermeasure approach. Table 6, shows the impact of attacks in the dataset on different SDN layers. It can be seen that the majority of the attacks in the dataset might cause damage to the SDN controller. The centralized control element displays the main differences

TABLE 5: Total Number of Data Instances and its Size.

Data Group	Traffic Distribution	Total Number of Instances	Total %	PCAP Size
Normal Group	Skype, Facebook, File Transfer, Youtube, Email, DNS, Chat, Browsing	68424	68424 (19.90%)	3.58 GB
Metasploitable-2 Group	DDoS	73529	136743 (39.76%)	669 MB
	Probe	61757		
	DoS	1145		
	brute-force-attack	295		
	Exploitation (R2L)	17		
OVS Group	DoS	52471	138772 (40.34%)	1.21 GB
	DDoS	48413		
	Probe	36372		
	brute-force-attack	1110		
	Web_attack	192		
	Botnet	164		

between the SDN and the conventional networks. In the conventional network, any attack can affect only one portion of the network, probably related to one vendor without interrupting the whole network services. However, any damage to the SDN controller can cause a severe impact on the entire system. Another concern, hardening the control messages in the southbound or northbound interface can threaten the whole network system. Therefore, the organizations should tackle the security issue in the early stages before implementing their SDN project. Any delay or wait to secure the network can cause service-affecting problems.

C. DATASET AVAILABILITY

The InSDN dataset is publicly available on http://iotseclab.ucd.ie/datasets/SDN/ (or http://aseados.ucd.ie/datasets/SDN/) [69] with the publishing of this paper.

VI. LIMITATIONS

1) Although SDN is applied in different network environments, the technology is still under development. Unfortunately, the previous history of SDN attacks is unknown. Therefore, in this work we act like the attacker and anticipate the weaknesses that he might be likely to strike.
2) The InSDN testbed was implemented using only ONOS SDN controller. The different types of functionalities in terms of security analysis for other controllers are ignored. However, authors in [70], [71] claim that the different controllers can have different security modeling, and therefore, different countermeasures.
3) SDN can be deployed in different network scales. It will be expected for SDN to support more devices and users more significant than the conventional network. Therefore, only one controller is not enough to cover all network nodes and users. For enterprise networks, there are probably several controllers connecting together through API interfaces such as east-bound and northbound interfaces. Unfortunately, due to the hardware constraints, the low scale topology with only one SDN controller was considered and implemented. However, using a single controller can perform well and achieve the purpose of optimal flow management [72]. In addition, obtaining the dataset using a single controller or multi controllers will not cause a big difference in methodology [35].
4) To generate more intrinsic data for SDN networks, the network topology should be created using physical devices. We tested various attacks and studied its impact on SDN layers by simulating the SDN network using virtual machines instead of real elements. We are planning to generate a more intrinsic dataset using physical topology with many connected devices.
5) The InSDN dataset assumes that all attacks are generated by high-level skill attackers. The threats, which come from misconfiguration or conflicting flow-tables in the switches are ignored.
6) One of the main limitations of the proposed dataset is a high-class imbalance. This problem can cause biasing of the IDS towards the majority class, causing high false alarm and low evaluation accuracy. However, there are many different techniques to solve the problem of imbalanced samples [73]–[76]. One of these techniques is applying a relabeling solution. Where two different methods can be used: (a) The high classes can be splatted to form more classes; (b) Merge two or more minority classes that share the same characteristics to create a new one class. As a result, the imbalance issue can be reduced, and prevalence ratio is effectively improved.

TABLE 6: Impact of Attacks in Dataset on the Different Elements of SDN.

Attack classes	Data Plane	Southbound Interface	Controller	Northbound Interface	Application Plane	Type of Threat
DoS	Yes	Yes	Yes	Yes	Yes	External
DDoS	Yes	Yes	Yes	Yes	Yes	Internal
Web Attacks	No	No	No	Yes	Yes	External
Brute Force Attack	Yes	No	Yes	Yes	Yes	External
Malware	Yes	Yes	Yes	Yes	Yes	External
Probe	Yes	Yes	Yes	Yes	Yes	External
Exploitation	No	No	Yes	Yes	Yes	External

VII. EXPERIMENTAL EVALUATION

This section analyzes eight supervised learning techniques to evaluate the usability and quality of InSDN dataset. The main objective is to demonstrate the quality of this dataset when it is used in the binary classification i.e. normal versus attack classes. Various performance indicators are used to evaluate the efficiency of employed supervised learning techniques, such as precision, recall, precision, F-score, and training time.

A. DATASET PRE-PROCESSING STEPS

The first phase before training the IDS models is to pre-process the dataset to make it more suitable for the training phase and avoid the overfitting problem. Few steps are taken for pre-processing the entering flows, as follows:

- The InSDN dataset contains the socket information such as Source IP, Destination IP, flow ID, etc. All socket features are removed to avoid the overfitting problem, where these features can be changed from network to network. The final dataset includes 77 various features, besides the traffic category.
- The features have different ranges, so they need to be standardized to restrict the scale of the values between 0 and 1.
- One-hot encoding scheme is used to convert the labeled string to numerical values. In this model, only binary classification is considered to classify the input data into malicious and normal group. Therefore, the normal and malicious strings are encoded into binary values of 0 and 1, respectively.

B. SDN SPECIFIC FEATURES

This section focuses on selecting the necessary features that can be directly obtained from the SDN network.

In SDN, only statistical features can be extracted from the SDN controller through OpenFlow calls to the SDN switches, (eg., flow duration, number of packets, number of bytes). In this manuscript, the same framework method of [77] is used to obtain the SDN specific features. These features can be directly extracted from the SDN controller through API queries or by the manual computation based on flow statistics information. Table 7 represents the corresponding mapping between derived features from the SDN environment to the InSDN dataset features. In addition, Table 8 shows extra features that can be calculated from the manual competition. The new features include the maximum, minimum, mean, and standard deviation of these values as well as the direction-specific features. These features are essential to define some particular attacks like botnet [77]. We selected a subset of 48 features from our dataset. While the previous method [77] used a subset of 50 features to train their learning model. However, they used the source IP, destination IP in their computation. These two attributes are excluded from the feature selection strategy, where IP addresses can be changed from network to network. Besides, the same IP address can be assigned to the attacker machine as well as the normal user. Thus, IP address is not able to distinguish between normal and attack traffic. Table 9 represents the total selected features for the SDN context from the proposed data.

C. MACHINE LEARNING ANALYSIS TECHNIQUES

This work uses eight common supervised learning algorithms to evaluate the quality of the InSDN dataset. Specifically, we employed three tree-based algorithms: a single Decision Tree (DT) [78], Random Forest (RF) [79], and Adaptive Boosting (AdaBoost) [80] learner. Besides, the k-nearest Neighbor classifier (Knn) [81], Naive Bayes (NB) [82], and two Support Vector Machines (SVM) [83] based method: linear kernel (lin-SVM) and a radial basis function kernel (rbf-SVM). In addition to the previous classifiers, a multi-layer perceptron model (MLP) is chosen in order to further evaluate the InSDN dataset. The hyper-parameters setting of MLP is described in the Table 10, while the default parameters are used in all the implemented algorithms. All learning classifiers are trained using the cross-validation technique with K=5, where the training and test data are splitted into 80% to 20%. In our experiments, there is no significant difference in terms of the accuracy between K = 5 and K = 10. In addition, using the larger K is subject to the computationally expensive and time consuming process, especially in large datasets. All the experiments were implemented in Python programming language using various libraries such as Keras, Scikit-Learn, and Tensorflow. Furthermore, all the experiments were performed using a workstation machine that has the following properties: Intel(R) UHD Graphics 620, I7-8650U CPU @ 1.90GHz (8 cores), 2.1GHz, Windows 10 pro 64-bit with 16 GB of RAM.

D. CLASSIFICATION METRICS

Using the complete accuracy does not yield precise comparisons [84], so we use the most important performance indicators to evaluate our proposed model, such as precision, recall,

TABLE 7: The Extracted Traffic Features from SDN Controller.

No.	Feature Description	SDN Derived Features	InSDN Dataset
1	Length of the connection	Duration	Flow Duration
2	Protocol_type	Protocol	Protocol
3	Max. expire time of flow	Hard_time_out	Flow use
4	Flow permanence time	Idle_time_out	Flow idle
5	Packets in bidirectional flow	Packets	Packets
6	Data bytes in bidirectional flow	Bytes_count	Bytes
7	Data bytes from source to dest.	Tx_packets	Src2dst_packets
8	Data bytes from dest. to source	Rx_packets	dst2src_packets

TABLE 8: The Extra Traffic Features.

No.	New Feature	InSDN Feature
1	Packet/s from source to dest. (PPS)	Packet rate (src2dst)
2	Packet/s from dest. to source	Packet rate (dst2src)
3	Inter-arrival time (IAT) (min, avg, max, std)	Inter time
4	Inter-arrival time from source to dest.(min, max, mean, std)	Inter time (src2dst)
5	Inter-arrival time from dest. to source (min, max, mean, std)	Inter time (dst2src)

TABLE 9: The subset Features for SDN environment.

No.	Attribute Name	No.	Attribute Name
1	Protocol	25	Fwd-IAT-Min
2	Flow-duration	26	Bwd-IAT-Tot
3	Tot-Fwd-Pkts	27	Bwd-IAT-Mean
4	Tot-Bwd-Pkts	28	Bwd-IAT-Std
5	TotLen-Fwd-Pkts	29	Bwd-IAT-Max
6	TotLen-Bwd-Pkts	30	Bwd-IAT-Min
7	Fwd-Pkt-Len-Max	31	Fwd-Header-Len
8	Fwd-Pkt-Len-Min	32	Bwd-Header-Len
9	Fwd-Pkt-Len-Mean	33	Fwd-Pkts/s
10	Fwd-Pkt-Len-Std	34	Bwd-Pkts/s
11	Bwd-Pkt-Len-Max	35	Pkt-Len-Min
12	Bwd-Pkt-Len-Min	36	Pkt-Len-Max
13	Bwd-Pkt-Len-Mean	37	Pkt-Len-Mean
14	Bwd-Pkt-Len-Std	38	Pkt-Len-Std
15	Flow-Byts/s	39	Pkt-Len-Var
16	Flow-Pkts/s	40	Pkt-Size-Avg
17	Flow-IAT-Mean	41	Active-Mean
18	Flow-IAT-Std	41	Active-Std
19	Flow-IAT-Max	43	Active-Max
20	Flow-IAT-Min	44	Active-Min
21	Fwd-IAT-Tot	45	Idle-Mean
22	Fwd-IAT-Mean	46	Idle-Std
23	Fwd-IAT-Std	47	Idle-Max
24	Fwd-IAT-Max	48	Idle-Min

TABLE 10: The hyper-parameters used in multi-layer perceptron approach.

Traffic Type	Detection Rate
Hidden nodes (HN)	2
Number of neurons	80, 100
Number of epoch	10
Batch size	100
Learning rate (LR)	0.001
Classification function	sigmoid
Activation function	relu

precision and F-score. These metrics are commonly used in intrusion detection systems and are defined as follows:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{1}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2}$$

$$\text{F-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

where True Positive (TP) and True Negative (TN) represent the values that are correctly predicted. In contrast, False Positives (FP) and False Negatives (FN) indicate missclassified events. Furthermore, we considered the training time to describe how long the classifier algorithm takes for training the whole data.

E. RESULTS AND DISCUSSION

This section discusses in detail the performance evaluation of the InSDN dataset.

1) Fully-featured version of the dataset.

Table 11 shows the performance of different classifiers using a fully-featured version of our dataset. It is clear that the overall score metrics are very high for DoS/DDoS and probe classes for all learner classifiers, while the U2R gives the poor performance metrics. This is because both DoS and Probe categories are commonly more different from normal traffic patterns [85]. In contrast, the U2R attack class has a high similarity to the normal connections. In addition, the size of U2R flow records is small compared to the normal

TABLE 11: Metrics Performance for the Fully-featured version of the dataset.

Attack Name	Matrices	Algorithm							
		KNN	NB	Adaboost	DT	RF	rbf-SVM	lin-SVM	MLP
1: DoS	Precision	0.998866	0.812257	0.999814	0.999381	0.999876	0.987335	0.981379	0.987201
	Recall	0.999134	0.977681	0.999897	0.999587	0.999835	0.997009	0.999072	0.997856
	F1-Score	0.999000	0.887325	0.999856	0.999484	0.999856	0.992148	0.990146	0.992171
	Training Time (Second)	71.657	2.389	379.479	10.525	61.695	852.874	202.16	243.349
2: DDoS	Precision	0.999967	0.999868	0.999967	0.999967	0.999992	0.999680	0.999639	0.999680
	Recall	0.999877	0.995621	0.999959	0.999943	0.999951	0.999885	0.999967	0.999951
	F1-Score	0.999922	0.997740	0.999963	0.999955	0.999971	0.999783	0.999803	0.999803
	Training Time	293.989	3.812	452.234	6.479	41.503	2825.451	209.679	1044.772
3: Password-Guessing Attack	Precision	0.990813	0.128224	0.997862	0.990787	0.996434	0.980682	0.981039	0.985154
	Recall	0.997865	0.997865	0.996441	0.995018	0.994306	0.614235	0.994306	0.986497
	F1-Score	0.994326	0.227247	0.997151	0.992898	0.995369	0.755361	0.987628	0.869021
	Training Time	42.8	1.532	237.381	4.196	22.712	33.165	25.72	324.482
4: Botnet	Precision	0.987952	1.000000	0.982036	0.981481	0.993865	0.987952	0.987952	1.0
	Recall	1.000000	1.000000	1.000000	0.969512	0.987805	0.500000	0.500000	0.634146
	F1-Score	0.993939	1.000000	0.990937	0.975460	0.990826	0.663968	0.663968	0.7847589
	Training Time	39.411	0.885	229.468	2.614	12.53	6.15	5.369	279.065
5: Web Attack	Precision	0.973404	0.016984	0.979381	0.963542	0.994764	0.988764	0.979167	0.944254
	Recall	0.953125	0.979167	0.989583	0.963542	0.989583	0.458333	0.489583	0.682292
	F1-Score	0.963158	0.033390	0.984456	0.963542	0.992167	0.626335	0.652778	0.686583
	Training Time	39.817	1.598	171.924	5.492	22.432	9.268	7.597	390.29
6: Probe	Precision	0.999705	0.992730	0.999745	0.999705	0.999857	0.998076	0.999429	0.999704
	Recall	0.999827	0.996362	0.999918	0.999817	0.999959	0.999338	0.999409	0.999449
	F1-Score	0.999766	0.994543	0.999832	0.999761	0.999908	0.998707	0.999419	0.999449
	Training Time	179.255	2.357	496.298	14.048	79.357	926.002	387.925	748.737
7: U2R	Precision	0.777778	0.061728	1.000000	0.800000	1.000000	0.923077	0.928571	0.916666
	Recall	0.823529	0.882353	0.823529	0.705882	0.529412	0.705882	0.764706	0.875
	F1-Score	0.800000	0.115385	0.903226	0.750000	0.692308	0.800000	0.838710	0.866666
	Training Time	87.702	2.931	170.534	1.973	11.532	3.231	2.962	530.091
8: Merged (all types)	Precision	0.999717	0.777388	0.999298	0.999898	0.999931	0.997692	0.999857	0.998854
	Recall	0.999793	0.985210	0.999123	0.999953	0.999942	0.999786	0.999898	0.999078
	F1-Score	0.999755	0.869047	0.999211	0.999926	0.999936	0.998738	0.999529	0.999858
	Training Time	1391.163	5.589	1147.662	35.631	231.436	17161.164	12161.164	806.736

TABLE 12: Metrics Performance for the SDN specific-featured version of the dataset.

Attack Name	Matrices	Algorithm							
		KNN	NB	Adaboost	DT	RF	rbf-SVM	lin-SVM	MLP
1: DoS	Precision	0.997237	0.744574	0.999443	0.998825	0.999732	0.826327	0.817605	0.979499
	Recall	0.997628	0.977186	0.999732	0.999175	0.999567	0.995008	0.994204	0.996410
	F1-Score	0.997432	0.845166	0.999588	0.999000	0.999649	0.902856	0.897298	0.980865
	Training Time (Second)	43.839	2.206	350.889	12.07	70.569	2271.213	789.381	163.3
2: DDoS	Precision	0.999934	0.990957	0.999975	0.999967	0.999992	0.999631	0.999557	0.999754
	Recall	0.999844	0.995662	0.999959	0.999951	0.999951	0.999672	0.999688	0.999852
	F1-Score	0.999889	0.993304	0.999967	0.999959	0.999971	0.999651	0.999623	0.999811
	Training Time	347.005	3.641	400.187	6.447	42.571	2521.642	113.666	695.941
3: Password-Guessing Attack	Precision	0.985229	0.108615	0.991398	0.987926	0.993534	0.964960	0.960000	0.976521
	Recall	0.949466	0.997865	0.984342	0.990036	0.984342	0.254804	0.290391	0.719573
	F1-Score	0.967017	0.195906	0.987857	0.988980	0.988917	0.403153	0.445902	0.829973
	Training Time	17.673	1.29	186.467	3.212	24.624	106.237	76.552	325.351
4: Botnet	Precision	0.959064	0.022356	0.993902	0.981928	0.993902	0.000000	0.000000	0.896984
	Recall	1.000000	1.000000	0.993902	0.993902	0.993902	0.000000	0.000000	0.496591
	F1-Score	0.979104	0.043733	0.993902	0.987879	0.993902	0.000000	0.000000	0.810501
	Training Time	17.97	1.223	177.872	2.118	12.809	8.889	9.174	219.566
5: Web Attack	Precision	0.984848	0.014902	1.000000	0.962162	1.000000	1.000000	0.000000	1.0
	Recall	0.677083	0.979167	0.968750	0.927083	0.942708	0.031250	0.000000	0.447503
	F1-Score	0.802469	0.029357	0.984127	0.944297	0.970509	0.060606	0.000000	0.617431
	Training Time	18.353	1.538	205.255	4.66	28.56	10.577	12.645	374.099
6: Probe	Precision	0.983743	0.907628	0.985157	0.987088	0.984909	0.923089	0.964200	0.975188
	Recall	0.991562	0.996810	0.993600	0.986243	0.993661	0.999134	0.999052	0.998552
	F1-Score	0.987637	0.950131	0.989361	0.986665	0.989266	0.959607	0.999052	0.977371
	Training Time	53.348	3.413	452.082	14.487	103.05	8244.774	4225.816	1520.646
7: U2R	Precision	0.800000	0.002616	1.000000	0.545455	1.000000	0.000000	0.000000	0.999751
	Recall	0.235294	0.882353	0.823529	0.705882	0.529412	0.000000	0.000000	0.0
	F1-Score	0.363636	0.005216	0.903226	0.615385	0.692308	0.000000	0.000000	0.066666
	Training Time	18.812	1.443	226.378	2.759	27.99	7.784	25.64	692.589
8: Merged (all types)	Precision	0.993991	0.951940	0.994115	0.994037	0.994269	0.963110	0.972192	0.988836
	Recall	0.995713	0.980498	0.997168	0.996177	0.996918	0.997622	0.995326	0.997818
	F1-Score	0.994851	0.966008	0.995640	0.995106	0.995592	0.980062	0.991338	0.993565
	Training Time	953.549	7.031	868.792	29.402	226.151	25837.86	152475.72	1820.143

flow in the same set. Furthermore, the overall performance of Adaboost and MLP is significantly high for all attack classes, but the training time is relatively long.

Recall and F1-Score on the botnet, web attack, and U2R classes are poor for both linear and RBF based SVM. Besides, the recall and F1-score metrics for rbf-SVM algorithm are low on the password brute-forcing attack type. Furthermore, the performance and training time of KNN, DT, and RF classifiers are reasonable for all attack classes. These algorithms succeeded in recognizing most of the attacks, but they have low scores in the U2R attack. In contrast, the NB classifier consumes less time in the learning and prediction stage compared to other classification algorithms, but its performance is significantly low for three attacks type, including Brute-Forcing, web attack and U2R classes. However, NB improved the results on botnet attack type compared to other algorithms. Another interesting finding is that the good results on the merged dataset might obfuscate poor performance on the less prevalent attack classes, as the majority of samples are for DoS/DDoS and probe attacks. Further, the training time is proportional to the data records, i.e. the training time is increased during the increase in the size of records. We can notice that the rbf-SVM had the most considerable training time for DoS/DDoS and probe, followed by MLP and Adaboost classifiers.

2) SDN specific version of the dataset.

Table 12 shows the performance of various models using SDN specific-featured version of the dataset. We can see that the Adaboost retains high-performance scores and stability for all attack classes, followed by DT and RF classifiers. However, the obtained scores on the U2R attack types are relativity small for DT and RF classifiers. In addition, Recall and F1-Score for KNN algorithm are relativity low for KNN on web attack and U2R attacks. We can find that the NB consistently had good scores on all metrics for DDoS and port attack classes, while its performance highly declined on botnet, web attack, and U2R classes. Furthermore, we noted a substantial declined in the performance of SVM on the botnet, password brute-forcing, web attack, and U2R attack classes. Where, the linear and RBF based SVM fail to identify any flow records for the botnet, web , and U2R attacks. While its recall and F1-Score metrics are very poor on password brute-forcing attack class. Although the stability of SVM (linear and RBF kernel) performance on DoS, DDoS, and probe attacks, its training time is effectively high, compared to the fully-version features of dataset. Furthermore, the recall score is decreased for MLP algorithm on password brute-forcing, botnet, and web attack types, while recall and F1-score are almost closed to zero for U2R class.

3) State of the Art Result Comparison

In this experiment, InSDN dataset is compared with four publicly available datasets (i.e, KDD'99, NSL-KDD, Kyoto and CICIDS 2017) by using six machine learning approaches, namely KNN, NB, Adaboost, DT, RF and rbf-SVM. As shown in Figure 7, it is clearly noticed that AB and RF classifiers performed well compared to other algorithms. In addition, DT, AB, and RF classifier performance remain the same over various datasets. However, KNN, NB, and SVM-rbf performance fluctuate across various datasets. This implies the power of DT, AB, and RF to detect the new attacks.

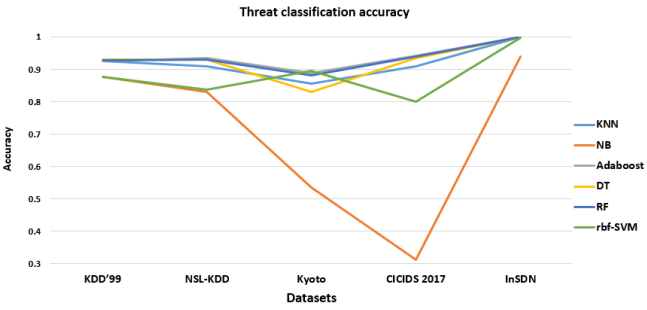

FIGURE 7: Performance of classification algorithms in term of global detection rate. We estimate our dataset accuracy compared with other publicly available datasets based on the proposed work in [86]

VIII. CONCLUSION

This paper investigated the challenging problem related to the dataset availability in the SDN environment. We proposed a new SDN dataset: InSDN, to solve some of the inherent problems in legacy datasets. We considered different attack scenarios that represent the real-world scenarios, and discussed the impact of the generated attacks on the different SDN elements. We can observe that the SDN can also be afflicted with the popular network attacks. However, the SDN network is more sensitive to malicious traffic than the conventional environments. In the conventional network, any attacks can only affect the portion of the network almost for the same vendor without bringing down the entire network. However, in the SDN environment, the compromised switches or end-users can flood the SDN controller, causing damage for the whole network.

In the near future, we will extend this work and create a more intrinsic dataset generated from large-scale networks. Moreover, we will consider new attack categories for the best representative of existing real-world networks.

REFERENCES

[1] H. Z. Jahromi and D. T. Delaney, "An application awareness framework based on sdn and machine learning: Defining the roadmap and challenges," in *2018 10th International Conference on Communication Software and Networks (ICCSN)*. IEEE, 2018, pp. 411–416.

[2] Statista.com, "Software-defined networking (sdn) market size worldwide from 2013 to 2021," [Online]. Available: https://www.statista.com/statistics/468636/global-sdn-market-size, [Accessed: 03-February-2020].

[3] A. Prakash and R. Priyadarshini, "An intelligent software defined network controller for preventing distributed denial of service attack," in *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*. IEEE, 2018, pp. 585–589.

[4] D. Li, C. Yu, Q. Zhou, and J. Yu, "Using svm to detect ddos attack in sdn network," in *IOP Conference Series: Materials Science and Engineering*, vol. 466, no. 1. IOP Publishing, 2018, p. 012003.

[5] 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.

[6] M. Myint Oo, S. Kamolphiwong, T. Kamolphiwong, and S. Vasupongayya, "Advanced support vector machine-(asvm-) based detection for distributed denial of service (ddos) attack on software defined networking (sdn)," *Journal of Computer Networks and Communications*, vol. 2019, 2019.

[7] T. Hurley, J. E. Perdomo, and A. Perez-Pons, "Hmm-based intrusion detection system for software defined networking," in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2016, pp. 617–621.

[8] A. S. da Silva, J. A. Wickboldt, L. Z. 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.

[9] A. Divekar, M. Parekh, V. Savla, R. Mishra, and M. Shirole, "Benchmarking datasets for anomaly-based network intrusion detection: Kdd cup 99 alternatives," in *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*. IEEE, 2018, pp. 1–8.

[10] L. Bontemps, V. Cao, J. McDermott, and N.-A. Le-Khac, "Collective anomaly detection based on long short-term memory recurrent neural networks," *In: Dang T., Wagner R., Küng J., Thoai N., Takizawa M., Neuhold E. (eds) Future Data and Security Engineering. FDSE 2016. Lecture Notes in Computer Science, vol 10018. Springer, Cham*, 2016.

[11] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*. IEEE, 2009, pp. 1–6.

[12] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," *ACM Transactions on Information and System Security (TISSEC)*, vol. 3, no. 4, pp. 262–294, 2000.

[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," in *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*. IEEE, 2016, pp. 258–263.

[14] T. A. Tang, L. Mhamdi, D. McLernon, and M. Zaidi, "Deep recurrent neural network for intrusion detection in sdn-based networks," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*. IEEE, 2018, pp. 202–206.

[15] J. Song, H. Takakura, and Y. Okabe, "Description of kyoto university benchmark data," *Available at link: http://www. takakura. com/Kyoto_data/BenchmarkData-Description-v5. pdf [Accessed on 15 March 2016]*, 2006.

[16] W. Haider, J. Hu, J. Slay, B. P. Turnbull, and Y. Xie, "Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling," *Journal of Network and Computer Applications*, vol. 87, pp. 185–192, 2017.

[17] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *computers & security*, vol. 31, no. 3, pp. 357–374, 2012.

[18] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." in *ICISSP*, 2018, pp. 108–116.

[19] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset," *Future Generation Computer Systems*, vol. 100, pp. 779–796, 2019.

[20] R. Panigrahi and S. Borah, "A detailed analysis of cicids2017 dataset for designing intrusion detection systems," *International Journal of Engineering & Technology*, vol. 7, no. 3.24, pp. 479–482, 2018.

[21] Canadian Institute of Cybersecurity, "Cse-cic-ids2018," 2018. [Online]. Available: https://www.unb.ca/cic/datasets/ids2018.html, [Accessed: 10-February-2020].

[22] J. Wang and I. C. Paschalidis, "Botnet detection based on anomaly and community detection," *IEEE Transactions on Control of Network Systems*, vol. 4, no. 2, pp. 392–404, 2016.

[23] C. Yin, Y. Zhu, S. Liu, J. Fei, and H. Zhang, "An enhancing framework for botnet detection using generative adversarial networks," in *2018 International Conference on Artificial Intelligence and Big Data (ICAIBD)*. IEEE, 2018, pp. 228–234.

[24] A. H. Lashkari, A. F. A. Kadir, L. Taheri, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark android malware datasets and classification," in *2018 International Carnahan Conference on Security Technology (ICCST)*. IEEE, 2018, pp. 1–7.

[25] S. Garcia, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *computers & security*, vol. 45, pp. 100–123, 2014.

[26] M. Ring, D. Landes, and A. Hotho, "Detection of slow port scans in flow-based network traffic," *PloS one*, vol. 13, no. 9, 2018.

[27] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, 2019.

[28] H. Hindy, D. Brosset, E. Bayne, A. Seeam, C. Tachtatzis, R. Atkinson, and X. Bellekens, "A taxonomy and survey of intrusion detection system design techniques, network threats and datasets," *arXiv preprint arXiv:1806.03517*, 2018.

[29] M. Conti, A. Gangwal, and M. S. Gaur, "A comprehensive and effective mechanism for ddos detection in sdn," in *2017 IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE, 2017, pp. 1–8.

[30] R. Santos, D. Souza, W. Santo, A. Ribeiro, and E. Moreno, "Machine learning algorithms to detect ddos attacks in sdn," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 16, 2020.

[31] T. Huang, F. R. Yu, C. Zhang, J. Liu, J. Zhang, and Y. Liu, "A survey on large-scale software defined networking (sdn) testbeds: Approaches and challenges," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 2, pp. 891–917, 2016.

[32] R. Braga, E. Mota, and A. Passito, "Lightweight ddos flooding attack detection using nox/openflow," in *IEEE Local Computer Network Conference*. IEEE, 2010, pp. 408–415.

[33] P. Amaral, J. Dinis, P. Pinto, L. Bernardo, J. Tavares, and H. S. Mamede, "Machine learning in software defined networks: Data collection and traffic classification," in *2016 IEEE 24th International Conference on Network Protocols (ICNP)*. IEEE, 2016, pp. 1–5.

[34] G. A. Ajaeiya, N. Adalian, I. H. Elhajj, A. Kayssi, and A. Chehab, "Flow-based intrusion detection system for sdn," in *2017 IEEE Symposium on Computers and Communications (ISCC)*. IEEE, 2017, pp. 787–793.

[35] H. Polat, O. Polat, and A. Cetin, "Detecting ddos attacks in software-defined networks through feature selection methods and machine learning models," *Sustainability*, vol. 12, no. 3, p. 1035, 2020.

[36] M. S. Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Machine-learning techniques for detecting attacks in sdn," in *2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*. IEEE, 2019, pp. 277–281.

[37] K. Benzekki, A. El Fergougui, and A. Elbelrhiti Elalaoui, "Software-defined networking (sdn): a survey," *Security and communication networks*, vol. 9, no. 18, pp. 5803–5833, 2016.

[38] M. S. Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Ddosnet: A deep-learning model for detecting network attacks," *arXiv preprint arXiv:2006.13981*, 2020.

[39] A. Dawoud, S. Shahristani, and C. Raun, "Software-defined network security: Breaks and obstacles," *Networks of the Future: Architectures, Technologies, and Implementations*, p. 89, 2017.

[40] D. Kreutz, F. M. Ramos, and P. Verissimo, "Towards secure and dependable software-defined networks," in *Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking*, 2013, pp. 55–60.

[41] H. Cheng, J. Liu, J. Mao, M. Wang, J. Chen, and J. Bian, "A compatible openflow platform for enabling security enhancement in sdn," *Security and Communication Networks*, vol. 2018, 2018.

[42] J. Benabbou, K. Elbaamrani, and N. Idboufker, "Security in openflow-based sdn, opportunities and challenges," *Photonic Network Communications*, vol. 37, no. 1, pp. 1–23, 2019.

[43] V. Moorthy, R. Venkataraman, and T. R. Rao, "Security and privacy attacks during data communication in software defined mobile clouds," *Computer Communications*, vol. 153, pp. 515–526, 2020.

[44] S. Lee, J. Kim, S. Woo, C. Yoon, S. Scott-Hayward, V. Yegneswaran, P. Porras, and S. Shin, "A comprehensive security assessment framework for software-defined networks," *Computers & Security*, vol. 91, p. 101720, 2020.

[45] J. C. C. Chica, J. C. Imbachi, and J. F. Botero, "Security in sdn: A comprehensive survey," *Journal of Network and Computer Applications*, p. 102595, 2020.

[46] M. P. Singh and A. Bhandari, "New-flow based ddos attacks in sdn: Taxonomy, rationales, and research challenges," *Computer Communications*, 2020.

[47] M. Liyanage, A. Braeken, A. Jurcut, M. Ylianttila, and A. Gurtov, "Secure communication channel architecture for software defined mobile networks," *Computer Networks*, vol. 114, pp. 32–50, 2017.

[48] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (ddos) attack dataset and taxonomy," in *2019 International Carnahan Conference on Security Technology (ICCST)*. IEEE, 2019, pp. 1–8.

[49] E. Vasilomanolakis, C. G. Cordero, N. Milanov, and M. Mühlhäuser, "Towards the creation of synthetic, yet realistic, intrusion detection datasets," in *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2016, pp. 1209–1214.

[50] G. Creech and J. Hu, "Generation of a new ids test dataset: Time to retire the kdd collection," in *2013 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2013, pp. 4487–4492.

[51] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow *et al.*, "Onos: towards an open, distributed sdn os," in *Proceedings of the third workshop on Hot topics in software defined networking*, 2014, pp. 1–6.

[52] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, 2010, pp. 1–6.

[53] "Mininet – an instant virtual network on your laptop (or other pc)." available at http://mininet.org/, Accessed 24 Jen 2020.

[54] K. K. Karmakar, V. Varadharajan, and U. Tupakula, "Mitigating attacks in software defined networks," *Cluster Computing*, vol. 22, no. 4, pp. 1143–1157, 2019.

[55] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks," *IEEE communications surveys & tutorials*, vol. 15, no. 4, pp. 2046–2069, 2013.

[56] Symantec, "Internet security threat report," *Tech. rep.. Symantec., [Online]. Available: https://symantec-enterprise-blogs.security.com/*, 2018.

[57] N. Sharma and S. Mukherjee, "A novel multi-classifier layered approach to improve minority attack detection in ids," *Procedia Technology*, vol. 6, pp. 913–921, 2012.

[58] M. Dhawan, R. Poddar, K. Mahajan, and V. Mann, "Sphinx: Detecting security attacks in software-defined networks." in *Ndss*, vol. 15, 2015, pp. 8–11.

[59] S. Hong, L. Xu, H. Wang, and G. Gu, "Poisoning network visibility in software-defined networks: New attacks and countermeasures." in *NDSS*, vol. 15, 2015, pp. 8–11.

[60] C. Yoon, S. Lee, H. Kang, T. Park, S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "Flow wars: Systemizing the attack surface and defenses in software-defined networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 6, pp. 3514–3530, 2017.

[61] M. Brooks and B. Yang, "A man-in-the-middle attack against opendaylight sdn controller," in *Proceedings of the 4th Annual ACM Conference on Research in Information Technology*, 2015, pp. 45–49.

[62] Y. Tseng, F. Naït-Abdesselam, and A. Khokhar, "A comprehensive 3-dimensional security analysis of a controller in software-defined networking," *Security and Privacy*, vol. 1, no. 2, p. e21, 2018.

[63] 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*, 2019.

[64] F. Khellah, "Control plane packet-in arrival rate analysis for denial-of-service saturation attacks detection and mitigation in software-defined networks," *Arabian Journal for Science and Engineering*, vol. 44, no. 11, pp. 9349–9362, 2019.

[65] R. U. Rasool, U. Ashraf, K. Ahmed, H. Wang, W. Rafique, and Z. Anwar, "Cyberpulse: A machine learning based link flooding attack mitigation system for software defined networks," *IEEE Access*, vol. 7, pp. 34 885–34 899, 2019.

[66] X. Ma, J. Li, Y. Tang, B. An, and X. Guan, "Protecting internet infrastructure against link flooding attacks: A techno-economic perspective," *Information Sciences*, vol. 479, pp. 486–502, 2019.

[67] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and vpn traffic using time-related," in *Proceedings of the 2nd international conference on information systems security and privacy (ICISSP)*, 2016, pp. 407–414.

[68] A. H. Lashkari, G. Draper-Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of tor traffic using time based features." in *ICISSP*, 2017, pp. 253–262.

[69] "Insdn dataset," http://iotseclab.ucd.ie/datasets/SDN/(orhttp://aseados.ucd.ie/datasets/SDN/), August 2020.

[70] R. Khondoker, A. Zaalouk, R. Marx, and K. Bayarou, "Feature-based comparison and selection of software defined networking (sdn) controllers," in *2014 World Congress on Computer Applications and Information Systems (WCCAIS)*. IEEE, 2014, pp. 1–7.

[71] K. Phemius, M. Bouet, and J. Leguay, "Disco: Distributed multi-domain sdn controllers," in *2014 IEEE Network Operations and Management Symposium (NOMS)*. IEEE, 2014, pp. 1–4.

[72] M. Alsaeedi, M. M. Mohamad, and A. A. Al-Roubaiey, "Toward adaptive and scalable openflow-sdn flow control: A survey," *IEEE Access*, vol. 7, pp. 107 346–107 379, 2019.

[73] C. Mera and J. W. Branch, "A survey on class imbalance learning on automatic visual inspection," *IEEE Latin America Transactions*, vol. 12, no. 4, pp. 657–667, 2014.

[74] S. Wang and X. Yao, "Multiclass imbalance problems: Analysis and potential solutions," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 4, pp. 1119–1130, 2012.

[75] R. Longadge and S. Dongre, "Class imbalance problem in data mining review," *arXiv preprint arXiv:1305.1707*, 2013.

[76] S. M. Abd Elrahman and A. Abraham, "A review of class imbalance problem," *Journal of Network and Innovative Computing*, vol. 1, no. 2013, pp. 332–340, 2013.

[77] P. Krishnan, S. Duttagupta, and K. Achuthan, "Varman: Multi-plane security framework for software defined networks," *Computer Communications*, vol. 148, pp. 215–239, 2019.

[78] N. Frosst and G. Hinton, "Distilling a neural network into a soft decision tree," *arXiv preprint arXiv:1711.09784*, 2017.

[79] M. Belgiu and L. Drăguţ, "Random forest in remote sensing: A review of applications and future directions," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 114, pp. 24–31, 2016.

[80] A. J. Wyner, M. Olson, J. Bleich, and D. Mease, "Explaining the success of adaboost and random forests as interpolating classifiers," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 1558–1590, 2017.

[81] S. Zhang, X. Li, M. Zong, X. Zhu, and R. Wang, "Efficient knn classification with different numbers of nearest neighbors," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 5, pp. 1774–1785, 2017.

[82] S.-C. Chu, T.-K. Dao, J.-S. Pan *et al.*, "Identifying correctness data scheme for aggregating data in cluster heads of wireless sensor network based on naive bayes classification," *EURASIP Journal on Wireless Communications and Networking*, vol. 2020, no. 1, pp. 1–15, 2020.

[83] S. M. H. Bamakan, H. Wang, T. Yingjie, and Y. Shi, "An effective intrusion detection framework based on mclp/svm optimized by time-varying chaos particle swarm optimization," *Neurocomputing*, vol. 199, pp. 90–102, 2016.

[84] G. Karatas, O. Demir, and O. K. Sahingoz, "Increasing the performance of machine learning-based idss on an imbalanced and up-to-date dataset," *IEEE Access*, vol. 8, pp. 32 150–32 162, 2020.

[85] M. S. Elsayed, N.-A. Le-Khac, S. Dev, and A. D. Jurcut, "Detecting abnormal traffic in large-scale networks," *arXiv preprint arXiv:2008.05791*, 2020.

[86] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41 525–41 550, 2019.

MAHMOUD SAID ELSAYED received the B.E. degree in electronics and communication engineering from Zagazig University, Egypt in 2007, the M.E. degrees in Information Security from Nile University, Egypt, in 2018. He has worked for several years in the industry through Huawei and IBM Co. in area of Computer Network and Security. He is currently pursuing the Ph.D. degree at the School of Computer Science, UCD, Dublin, Ireland. His research interests involve computer networks, network security, deep learning, cloud computing.

DR. NHIEN-AN LE-KHAC (M'06) Lecturer at the School of Computer Science (CS), University College Dublin (UCD), Ireland. He is currently the Programme Director of UCD MSc programme in Forensic Computing and Cybercrime Investigation, an international programme for the law enforcement officers specialising in cybercrime investigations. To date, more than 1000 students from 60 countries in 5 continents have graduated from this FCCI programme. He is also the co-founder of UCD-GNECB Postgraduate Certificate in fraud and e-crime investigation. He was a Research Fellow in Citibank, Ireland (Citi). He obtained his PhD. in Computer Science in 2006 at the Institut National Polytechnique de Grenoble (INPG), France. His research interest spans the area of Cybersecurity and Digital Forensics, Machine Learning for Security, Fraud and Criminal Detection, Cloud Security and Privacy and High Performance computing. Since 2013, he has collaborated on many research projects as a principal/co-PI/funded investigator. He has published more than 150 scientific papers in peer-reviewed journal and conferences in related research fields. He is an active chair as well as a reviewer for many key conferences and journals in related disciplines.

DR. ANCA JURCUT Assistant Professor at the School of Computer Science, UCD. She received a bachelor of mathematics and computer science from West University of Timisoara, Romania (2007) and a Ph.D from University of Limerick, Ireland (2013). From 2008 to 2013, she was a research assistant with the Data Communication Security Laboratory at University of Limerick, and from 2013 to 2015, she was working as a postdoctoral researcher in the Department of Electronic and Computer Engineering at the University of Limerick and as a software engineer at IBM, Ireland. Since 2015, she has been an assistant professor with the School of Computer Science, University College Dublin,Ireland. Her research interests focuses on network and data security, security for internet of things (IoT), security protocols, formal verification techniques and applications of blockchain technologies in cybersecurity.

APPENDIX. DATA COLLECTION FEATURES.

TABLE 13: The list of Entire Features in the InSDN Dataset.

Network-identifiers attributes			
Feature	Feature name	Description	Type
F1	Flow-id	ID of the flow	C
F2	Src-IP	Source IP address	C
F3	Src-Port	Source port number	C
F4	Dst-IP	Destination IP address	C
F5	Dst-Port	Destination port number	C
F6	Protocol-Type	Type of protocol, e.g., tcp, udp, etc.	D
F7	Timestamp	Timestamp	C
Byte-based attributes			
F8	Fwd-Header-Len	Total bytes used for headers in the forward direction	C
F9	Bwd-Header-Len	Total bytes used for headers in the backward direction	C
Packet-based attributes			
F10	Tot-Fwd-Pkts	Total packets in the forward direction	C
F11	Tot-Bwd -Pkts	Total packets in the backward direction	C
F12	TotLen-Fwd-Pkts	Total size of packet in forward direction	C
F13	TotLen-Bwd-Pkts	Total size of packet in backward direction	C
F14	Fwd-Pkt-Len (Min, Mean, Max, Std)	Min, Mean, Max, and standard deviation of the size of packet in forward direction	C
F15	Bwd-Pkt-Len (Min, Mean, Max, Std)	Min, Mean, Max, and standard deviation of the size of packet in backward direction	C
F16	Pkt-Len (Min, Mean, Max, Var, Std)	Min, Mean, Max, Var and standard deviation of the length of a packet	C
F17	Pkt-Size-Avg	Average size of packet	C
Interarrival Times attributes			
F18	Duration	Duration of the flow in Microsecond	C
F19	Flow-IAT (Min, Mean, Max, Std)	Min, Mean, Max, and standard deviation of the time between two packets sent in the flow	C
F20	Fwd-IAT (Tot, Min, Mean, Max, Std)	Tot, Min, Mean, Max, and standard deviation of the time between two packets sent in the forward direction	C
F21	Bwd-IAT (Tot, Min, Mean, Max, Std)	Tot, Min, Mean, Max, and standard deviation of the Time between two packets sent in the backward direction	C
Flow timers attributes			
F22	Active-Time (Min, Mean, Max, Std)	Min, Mean, Max, Standard deviation of the time flow was active before becoming idle	C
F23	Idle (Min, Mean, Max, Std)	Min, Mean, Max, Standard deviation time flow was idle before becoming active	C
Flag-based attributes			
F24	Fwd-PSH-Flags	Number of times the PSH flag was set in packets travelling in the forward direction	D
F25	Bwd-PSH-Flags	Number of times the PSH flag was set in packets travelling in the backward direction (0 for UDP)	D
F26	Fwd-URG-Flags	Number of times the URG flag was set in packets travelling in the forward direction (0 for UDP)	D
F27	Bwd-URG-Flags	Number of times the URG flag was set in packets travelling in the backward direction (0 for UDP)	D
F28	FIN-Flag-Cnt	Number of packets with FIN	D
F29	SYN-Flag-Cnt	Number of packets with SYN	D
F30	RST-Flag-Cnt	Number of packets with RST	D
F31	PSH-Flag-Cnt	Number of packets with PUSH	D
F32	ACK-Flag-Cnt	Number of packets with ACK	D
F33	URG-Flag-Cnt	Number of packets with URG	D
F34	CWE-Flag-Cnt	Number of packets with CWE	D
F35	ECE-Flag-Cnt	Number of packets with ECE	D
Flow-based attributes			
F36	Down/Up-Ratio	Download and upload ratio	D
F37	Fwd-Seg-Size-Avg	Average size observed in the forward direction	C
F38	Bwd-Seg-Size-Avg	Average number of bytes bulk rate in the forward direction	C
F39	Fwd-Byts/b-Avg	Average number of bytes bulk rate in the forward direction	D
F40	Fwd-Pkts/b-Avg	Average number of packets bulk rate in the forward direction	D
F41	Fwd-Blk-Rate-Avg	Average number of bulk rate in the forward direction	D
F42	Bwd-Byts/b-Avg	Average number of bytes bulk rate in the backward direction	D
F43	Bwd-Pkts/b-Avg	Average number of packets bulk rate in the backward direction	D
F44	Bwd-Blk-Rate-Avg	Average number of bulk rate in the backward direction	D
F45	Init-Fwd-Win-Byts	The total number of bytes sent in initial window in the forward direction	C
F46	Init-Bwd-Win-Byts	The total number of bytes sent in initial window in the backward direction	C
F47	Fwd-Act-Data-Pkts	Count of packets with at least 1 byte of TCP data payload in the forward direction	C
F48	Fwd-Seg-Size-Min	Minimum segment size observed in the forward direction	C
F49	Flow-Byts/s	Number of flow bytes per second	C
F50	Flow-Pkts/s	Number of flow packets per second	C
F51	Fwd-Pkts/s	Number of forward packets per second	C
F52	Bwd-Pkts/s	Number of backward packets per second	C
Subflow-based attributes			
F53	Subflow-Fwd-Pkts	The average number of packets in a sub flow in the forward direction	C
F44	Subflow-Fwd-Byts	The average number of bytes in a sub flow in the forward direction	C
F55	Subflow-Bwd-Pkts	The average number of packets in a sub flow in the backward direction	C
F56	Subflow-Bwd-Byts	The average number of bytes in a sub flow in the backward direction	C
C-Continuous, D-Discrete			