IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 6, 2019, accepted January 24, 2019, date of publication January 31, 2019, date of current version February 20, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2896783

Efficient Distributed Denial-of-Service Attack Defense in SDN-Based Cloud

TRUNG V. PHAN[1], (Student Member, IEEE), AND MINHO PARK[2], (Member, IEEE)
[1]Department of Information Communication, Materials, and Chemistry Convergence Technology, Soongsil University, Seoul 156-743, South Korea
[2]School of Electronic Engineering, Soongsil University, Seoul 156-743, South Korea

Corresponding author: Minho Park (mhp@ssu.ac.kr)

This work was supported by the Institute for Information and Communications Technology Promotion through the Korea Government software defined networking security technology development, under Grant MSIT 2018-0-00254.

ABSTRACT Software-defined networking (SDN) is the key outcome of extensive research efforts over the past few decades toward transforming the Internet infrastructure to be more programmable, configurable, and manageable. However, critical cyber-threats in the SDN-based cloud environment are rising rapidly, in which distributed denial-of-service (DDoS) attack is one of the most damaging cyber attacks. In this paper, we propose an efficient solution to tackle DDoS attacks in the SDN-based cloud environment. We first introduce a new hybrid machine learning model based on support vector machine and self-organizing map algorithms to improve the traffic classification. Then, we propose an enhanced history-based IP filtering scheme (*eHIPF*) to improve the attack detection rate and speed. Finally, we introduce a novel mechanism that combines both the hybrid machine learning model and the *eHIPF* scheme to make a DDoS attack defender for the SDN-based cloud environment. The testbed is implemented in an SDN-based cloud with service function chaining. Through practical experiments, the proposed DDoS attack defender is proven to outperform existing mechanisms for DDoS attack classification and detection. The comprehensive experiments conducted with various DDoS attack levels prove that the proposed mechanism is an effective, innovative approach to defend DDoS attacks in the SDN-based cloud.

INDEX TERMS Distributed denial-of-service attacks, machine learning, software defined networks, network function virtualization.

I. INTRODUCTION

In recent years, Software Defined Networking (SDN) [1] and Network Functions Virtualization (NFV) [2] have emerged as cloud computing technologies. SDN is an innovative network framework that can monitor and control network traffic by utilizing the control-data plane detachment. Meanwhile, NFV has been developed as a novel solution technology to design, deploy and control network services with much lower costs by separating the functions from physical network devices. In addition, Service Function Chaining (SFC) [3] technology, which is enabled by both SDN and NFV, was proposed to support a sequence of multiple service functions (e.g., Firewall, DPI, Load Balancing) to a specific network flow. SDN, NFV, SFC technologies and cloud platform (OpenStack [4]) assist and benefit from each other to make a future SDN-based cloud environment as shown in Figure 1.

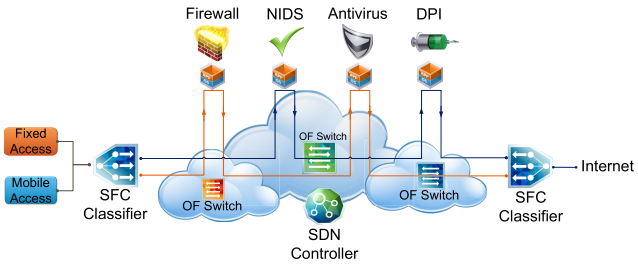

FIGURE 1. Future SDN-based cloud environment.

A. PROBLEM STATEMENTS

While SDN-based cloud is more advantageous in network traffic control and elastic resource management for a better cloud service in the future, it causes the vulnerability to Distributed Denial-of-Service (DDoS) attacks to both SFC operation and the cloud provider [5], [6]. The main purpose of DDoS attacks is to flood a victim with a massive traffic volume that is generated from botnets, which deplete computing

The associate editor coordinating the review of this manuscript and approving it for publication was Ghufran Ahmed.

resources and shut down the victim network systems. The reasons why DDoS attacks are tough challenges for the SDN-based cloud environment can be listed as follows

- First, every Software Defined Networking or OpenFlow-based network is now vulnerable to DDoS attacks because an OpenFlow switch is normally able to maintain from several up to a hundred thousands of flow-entries. However, when the network is under a large-scale DDoS attack, the flow number can rocket to some millions. This leads to not only the victim machine is affected, but also the SDN Controller and OpenFlow switches stop working due to resource exhaustion.
- Regarding the use of SFC in cloud environment, every service chain including a determined sequence of virtual network functions (VNFs) is fixed in a specific network path and may be public to internet users for commercial purposes. This can result in DDoS attackers flooding unwanted traffic to these SFCs to further degrade the public service quality or other reasons [7]. For instance, a service function chain consists of three VNFs such as: a Firewall, a DPI and a public service (e.g. Web). If this SFC is attacked by DDoS attackers, prior functions can be overloaded because of the rocketed attack traffic volume. In that case, there might be some actions from the cloud provider such as initiating more VNFs or randomly dropping incoming traffic to ensure the next VNFs are able to handle the service chain traffic. However, every SFC customer has a fixed resource allocation and provides users an acceptable quality of service depending on service types (Web, video and etc.) in terms of delay, packet loss rate, and so on. Therefore, this DDoS attack will reject legitimate user requests due to excessive resource consumption and traffic disruption.
- Moreover, if the small or medium-sized cloud providers do not provide any security solutions for their customers and themselves, some attack target domains in the cloud may become a traffic bottleneck when the traffic increases rapidly in a short period of time, and network devices or servers may go down in the worst case.

From mentioned issues, we can see that the damaging effects of DDoS attacks cause great difficulties to every network system not only in a legacy cloud network, but also SDN-based cloud environment. In addition, to the best of our best knowledge and as seen in [7] and [8], not many studies can effectively solve these grave problems.

B. OUR PROPOSAL

To resolve the serious issues given above and improve the robustness of the cloud system, we present a concrete proposal with a novel mechanism that monitors, checks and filters incoming traffic before forwarding to VNFs on the cloud.

In our proposal, we first introduce a novel combined machine learning algorithm to enhance the performance of classification in network traffic. It mainly takes advantages of two classification algorithms, Support Vector Machine (SVM) taking little time to produce outputs with a high accuracy and Self Organizing Map (SOM) making a reliable prediction based on their neurons, in order to minimize resource consumption while ensuring a high traffic classification performance. We then present an enhanced History-based IP Filtering scheme (*eHIPF*) to enhance the detection time of an abnormal source accessing the cloud system. Finally, we propose a novel security mechanism that combines both the hybrid machine learning model and the *eHIPF* scheme to tackle DDoS attacks in the SDN-based cloud, especially for SFC protection.

C. CONTRIBUTIONS

In summary, the major contributions of our research consist of the following:

- We introduce a new machine learning hybrid model for DDoS attack classification [9] based on Support Vector Machine (*SVM*) [10] and Self Organizing Map (*SOM*) [11], [12] algorithms to improve the performance of classification in network traffic.
- We propose an enhanced History-based IP Filtering scheme (*eHIPF*) in comparison with prior studies to improve the detection rate and speed in distinguishing DDoS attack's source IP addresses.
- We propose a novel security mechanism that combines both the hybrid machine learning model and the *eHIPF* scheme to produce an efficient DDoS attack defender in the SDN-based cloud.

The rest of the paper is structured as follows. Section II presents several related researches to our work. Section III first gives a brief introduction to a novel hybrid machine learning model, and describes the concept of Software Defined Networks and the Network Functions Virtualization technologies in cloud environment. Section IV mainly focuses on system analyses and practical design of the proposed security mechanism. Our experiments are conducted in Section V. Results and performance evaluation are presented in Section VI. Finally, conclusion and future researches are given in Section VII.

II. RELATED WORK

In the integrated SDN network environment, many studies [13], [14] proposed various solutions for Dos/DDoS attack detection and prevention recently.

Non-machine learning-based solutions also have been widely proposed to tackle DoS/DDoS attacks in SDN environment. For instance, FLOOD-GUARD [15] provides two modules: proactive flow rule analyzer and packet migration to preserve network policy enforcement and protect the controller from being overloaded. Van Trung *et al.* [16] introduce the usage of Fuzzy Interference System into DDoS attack prevention in the SDN. Reference [17] proposes a combined anomaly detection mechanism comprised of: (a) reduced data gathering with sampling, implemented

with the use of the sFlow protocol, (b) anomaly detection, implemented by entropy-based algorithm and (c) network-wide anomaly mitigation using OpenFlow, in which DDoS attacks is one of threats they tested using the proposed mechanism. Mehdi *et al.* [18] argue that the advent of Software Defined Networking provides a unique opportunity to effectively detect and contain network security problems in home and home office networks. Then, they present how four traffic anomaly detection (e.g. DDoS detection) algorithms can be implemented in an SDN context. A proposal, namely Fayaz [19], is introduced as a flexible and elastic DDoS defense system which addresses key challenges with respect to scalability, responsiveness, and adversary-resilience. Peng *et al.* [20], Dao *et al.*, and Goldstein [22] propose and conduct a DDoS detection engine by analyzing some historic datasets and finding hard thresholds to distinguish the attack sources from normal sources. In summary, non-machine learning-based solutions are effectively evaluated in DDoS detection and prevention in SDN environment. However, these approaches are quite complex in deployment and not adaptable to rapidly changes in network status, especially SDN-based networks. In other words, intelligent and adaptive detection and mitigation system is required for dealing with DDoS attacks in SDN-based networks.

Accordingly, several machine learning-based approaches [23] have been proposed to handle saturation attacks which can provide artificial intelligence in DDoS detection and mitigation. For example, Braga *et al.* [24] created a DDoS detection scheme using Self-Organizing Map (SOM) with 4 and 6 tuples of attributes, while Support Vector Machine classifier is applied to recognizing DDoS attack traffic in [25]. Phan *et al.* [26] propose a distributed scheme leveraging SOM algorithm to cope with flooding attacks. A deep learning based multi-vector DDoS detection system [27] is introduced in SDN-based environment. Meti *et al.* [28] proposed to utilize the Support Vector Machine classifier and the Neural Network classifier to detect the suspicious and harmful connections in the SDN controller. Overall, these proposed methods mainly base on one or a combination of machine learning algorithms in order to make a DDoS detection system in SDN-based networks. However, they focus on only intelligently DDoS detection and forget about the importance of other techniques (e.g. history-based) that can be applied along with their machine learning proposals in order to improve further processes such as mitigation and optimization.

From above analyses, this motivates us to propose a novel approach compared to previous studies, which is based on both machine learning and history-based techniques to efficiently tackle DDoS attacks in the SDN-based cloud environment.

III. BACKGROUND KNOWLEDGE

A. SDN-BASED CLOUD

The integration of SDN and NFV technologies is referred to as the SDN/NFV architecture [29], as shown in Figure 2. It includes NFV Orchestration, Controller platform, forwarding devices and servers. The traffic path is determined by the SDN controller using mainly OpenFlow protocol to communicate with forwarding devices (OpenFlow switch) to enforce policies from the control plane to data plane. Meanwhile, the NFV allows servers or cloud platform (e.g. OpenStack) to produce high-bandwidth and high-performance Network Functions without great cost. Hypervisors, which run on servers, majorly focus on supporting VMs that allow to operate Network Functions such as IDS, Firewall, Proxies.

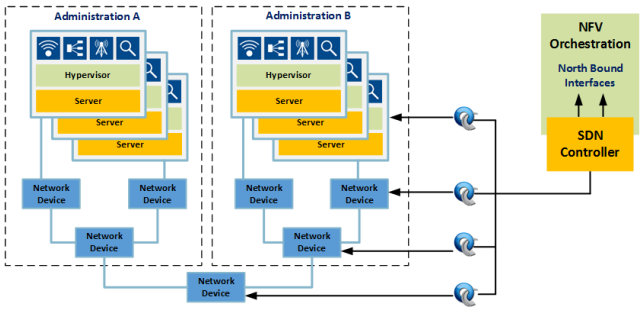

FIGURE 2. SDN-based cloud design.

The SDN controller [30]–[33] and the NFV orchestration are responsible for the logical control functions. The NFV orchestration system performs VNFs provisioning, and it is integrated with the SDN controller through interfaces or APIs. After considering the policy requirements and generating network topology from the Topology Management in the control platform, the Controller produces optimal function assignments and assigns the functions to certain VMs in the optimized path which can be known as a service chain [3]. The NFV orchestration system conducts a service function chain, and the controller instructs the traffic through a determined sequence of VMs by installing flow rules into forwarding devices.

B. DDOS ATTACKS IN SDN-BASED CLOUD

We briefly discuss DDoS attacks in traditional networks. In these network models, a wide range of approaches are used by attackers to attack victim [34], but we can summarize into two main types: bandwidth depletion attacks and resource depletion attacks. In the bandwidth depletion attacks, attackers flood a victim with unwanted traffic, which exhausts the victim network's bandwidth. This results in legal traffic not being able to access the victim network. For instance, UDP flooding, ICMP flooding or Smurf and Fraggle attacks [13]. With respect to the resource depletion attacks, attackers aim to send IP packets that are malformed or misuse the network protocol. Consequently, the victim suffers from resource exhaustion, and when the volume of connection is enough, the victim will stop working. TCP SYN flooding is a good illustration which bases on the three-handshake protocol between sender and receiver before opening a TCP connection.

TABLE 1. DDoS attacks in SDN-Based cloud.

Type I	Type II
ICMP flooding attack	TCP SYN attack
IGMP flooding attack	UDP flooding attack
Fraggle attack	PUSH + ACK attack
Smurf attack	Low and Slow rate attack

From the perspective of SDN, which is a flow rule-based network system, we also classify DDoS attacks into two major types as follows:

Type I: The main idea is to rely on the volume of packets or data coming from a source address. When the network system is under this type of DDoS attack, a prominent characteristics is that a source IP address connects to the victim network by generating one or two flows with a high level of the volume of packets in each flow. For example, ICMP flooding attack, Smurf and Fraggle attacks [13].

Type II: The second type bases on the volume of the number of flow to break down the victim network system. The basic idea is that a source IP address generates a large number of flows to a victim address in a short time (e.g. TCP SYN flooding [13]) and may keep these flows alive during the attack (e.g. low and slow rate DDoS attack [35]). This not only makes the victim, but also network devices such as OpenFlow switches, the SDN controller or VNFs to be crashed because of resource consumption (e.g. packet_in process, RAM and so on) [15], [35].

In conclusion, from a new point of view in the SDN-based environment and our analyses, these common DDoS attacks can be summarized as shown in Table 1.

C. A NOVEL HYBRID MACHINE LEARNING MODEL

In this section, we provide readers with a briefly discussion of the combination of Support Vector Machine (SVM) and Self Organizing Map (SOM). Interested readers can refer to our previous work [9] for more detailed information.

In [9] , we present a novel hybrid flow-based Distributed Denial-of-Service defender in Software Defined Networks, the core idea is to produce a two-algorithm combination that helps an intelligent security system enhance the accuracy in differentiating normal flows from abnormal flows during runtime. The SVM [10] performs as a high-speed classifier based on a hyperplane or set of hyperplanes in a high-dimensional space. In security problem solving, the SVM is also evaluated as a resource saving, high accurate classification algorithm or application that consumes low computational resources. However, some data points belong to a vague region or vague space for multiple dimensions which is limited by two margin lines or planes in the SVM algorithm. These vague data points are considered as suspicious points. In order to find their exact spaces, we take an unsupervised learning algorithm - SOM into account to recognize these data points with higher prediction results. By implementing our system, we proved that the proposed hybrid classification model outperforms original algorithms, and it can protect

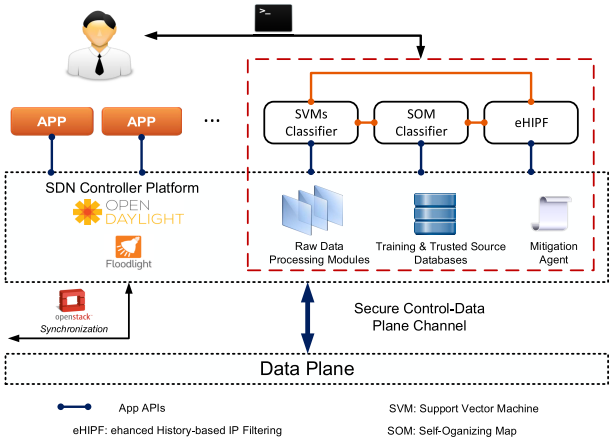

FIGURE 3. Conceptual Architecture.

the OpenFlow switches and the SDN controller from being overloaded under DDoS attacks.

Although the proposed solution in [9] already improved the classification performance of the traffic in SDN-based networks, there are some limitations. For instance, one company including several LAN networks only with some public IP addresses can access the Internet or public web services on the cloud and some PCs in LAN networks are bots of DDoS attackers. Then, the attackers send commands to their bots to send traffic to a targeted web server on the cloud. Unfortunately, many legal officers access to the web server at the same time, and this leads to one public IP may be used for both normal and attack request flows to the destination sever. Hence, the cloud security scheme may quickly detect attack flows and ban the IP address for a period of time. This makes normal users unable to access the web server. In addition, we note that many normal TCP flows behave like attack flows because of a very little packets at the beginning stage. As a result, the system may make bad decisions on legal requests and may apply strict polices (e.g. *delete* action [36]). This might also raise a concern in extra resource consumption due to the new flow installation process.

All things considered, this motivated us to propose new algorithms and mechanisms to overcome the mentioned limitations of our previous research and efficiently defend against DDoS attacks in the SDN-based cloud environment.

IV. SYSTEM ANALYSIS AND PRACTICAL DESIGN

To overcome the security problems related to DDoS attacks in previous sections, we introduce a novel, executable and practical framework in this section. First, we conduct a thorough logical analysis of the system, and we then illustrate how the proposed framework is designed with the main components in further detail.

A. SYSTEM ANALYSIS AND GOALS

In the SDN-based cloud, we suppose we are given following groups:

- Let $S = \{s_1, s_2, s_3, ..., s_x\}$ shows a group of all legitimate source IP addresses that accessed the SDN-based cloud in a specific period of h hours in nonattack condition, where $|S| = x$. This means the group S is updated after h hours as a loop because our proposed mechanism always considers every source IP as a potential DDoS attacker after h hours. Then, if a source IP overcomes security checkers (*SVM-k*, SOM and *eHIPF*) after some observation times, it will be considered as a legitimate source. Peng *et al.* [20], Dao *et al.*, and Goldstein [22] used a traced dataset and always set the considering time up to some days, and this not only makes a large database, but also is useless if there are lots of sources that only accessed just once or twice. Therefore, it is reasonable, if we only consider our history database in h hours.
- Let $T = \{t_1, t_2, t_3, ..., t_y\}$ presents a group of all trusted source IP addresses provided by the SDN-based cloud provider, where $|T| = y$. This group is updated by a *Trusted Source Database* (*TSD*) if there are some changes from the cloud administrator.
- Let $A = \{as_1, as_2, as_3, ..., as_z\}$ illustrates a group of all source IP addresses except trusted IPs that appeared on the cloud system in DDoS attack time, where $|A| = z$.

From these above groups, we have the number of source IP addresses in the SDN-based cloud system $srcIP_{num}$ as follows

$$\begin{cases} srcIP_{num} = x + y, & Nonattack, \\ srcIP_{num} = z + y \gg x + y, & DDoS\ attacks. \end{cases} \quad (1)$$

At first, group A can be seen as group S in the normal condition when only legal users access the cloud system. However, when the system is under DDoS attacks, the attackers always use a vast number of random spoofed source IP addresses to send attack traffic to a predetermined victim. Under the SDN perspective, the victim is an SFC (NIDS, Firewall, Web and etc) that provides a public service, such as Web or FTP services. Hence, the number of source IP address increases dramatically, which means that the value of z fires a rocket in a short time. Therefore, *our first goal* is to keep the value of $srcIP_{num}$ as small as $(x+y)$ possible in both normal and attack conditions. In other words, our introduced mechanism detects attack sources as quickly as possible, and distinguishes them from normal and trusted sources.

In addition, we assume that the incoming traffic follows Poisson distribution [37] in the normal traffic condition with a parameter λ, revealing the rate of average packet arriving, and the SDN-based cloud serves n customers for their SFCs in the cloud system. For each customer, they always have to negotiate with the cloud provider for the network resource (Bandwidth) from the cloud gateway to their SFCs and the VNF resources (CPU, RAM memory, Disk space, ...) before launching their services, which depends on the cloud infrastructure and the customer's budget. Let B_m (bps) is the upper bandwidth bound inside the cloud of the mth customer ($m \leq n$). Normally, if there are no DDoS attacks, the actual transfer bandwidth is always less than B_m (bps) and the VNFs are not overloaded due to the customer having their own stable users. However, when a DDoS attack happens, if the cloud provider forwards all incoming traffic without filtering to SFCs of their customer, it results in a rocket in bandwidth reaching B_m (bps). Hence, legitimate and trusted source IP addresses are unable to access the destination service due to some overloaded VNFs in their SFC. Accordingly, a demanding requirement for the SDN-based cloud provider is to provide a security mechanism for customer traffic before forwarding the filtered traffic to their SFCs to ensure the SFC quality of service (QoS) [38]. This leads to *our second goal* is to keep the traffic to the mth customer's SFC always being less or equal than the B_m (bps) from the border OpenFlow switches. This means that our proposed solution has to prevent abnormal traffic from accessing customer SFCs as much as possible inside the cloud.

B. SYSTEM DESIGN

To achieve the two mentioned goals in Section IV-A to face with saturation attacks while guaranteeing the SFC Quality of Service, we introduce a novel combined scheme among *SVM* classifier, *SOM* classifier and *eHIPF* mechanism to defend against DDoS attacks in the SDN-based cloud environment. This combined operation includes extension modules that can be implemented and distributed both in the SDN application plane and cloud controller platform (OpenStack [4]). We suggest that these modules could be placed at a dedicated security server in real deployment in order to reduce the SDN controller processing load. For convenience, however, in this work we design modules including *SVM*, *SOM* and *eHIPF* and locate these modules in the SDN application layer, just some modules are placed in the OpenStack controller for synchronization and further actions as shown in Figure 3. We build some functional modules: Raw Data Processing, two databases (Training and Trusted Source), Mitigation Agent, Statistic Sender and Update Agent, which are placed in the SDN control plane. Note that, the proposed scheme can be implemented in a distributed manner of nowadays cloud computing, in which databases can be shared among distributed agents to enhance anomaly detection performance. We next explain in more detail in following sections.

1) RAW DATA PROCESSING

As can be seen in Figure 4, we illustrate how the introduced modules connect and make a workable defending system. First, in order to get data from data plane, we run a Statistic Sender that sends frequently request messages [36] to OpenFlow switch and wait for response messages. After receiving the response data, OpenFlow Channel forwards them to the Raw Data Processing module. These data (raw data) are processed by Flow Collector, Feature Extractor and Traffic Classifier modules before going to an appropriate *SVM-k* classifier.

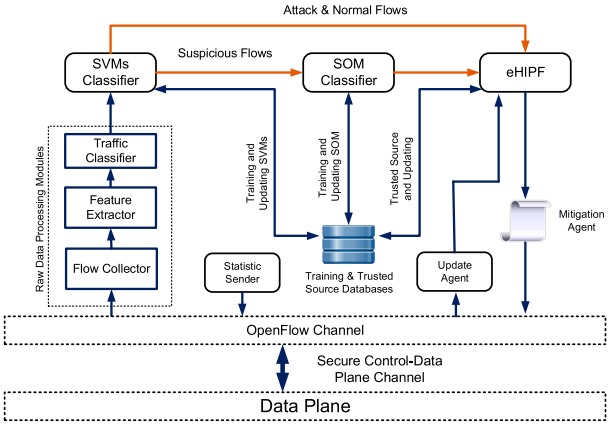

FIGURE 4. Detailed Architecture synchronized OpenStack Controller .

- *Flow Collector*: This module is a simple module running in the SDN controller, and it simply receives *StatsResponse* messages [36] in a preset period of time.
- *Feature Extractor*: It extracts flow information from *StatsResponse* messages to take out several attributes of a flow. Two of them are inputs of the *SVM-k* and the *SOM* processes four attributes (shown in Figure 5).
- *Traffic Classifier*: This module is accountable to transfer the attributes of a flow to the corresponding *SVM-k*. For example, a flow that protocol field is ICMP, flow information will be sent to the *SVM-ICMP* for classification.

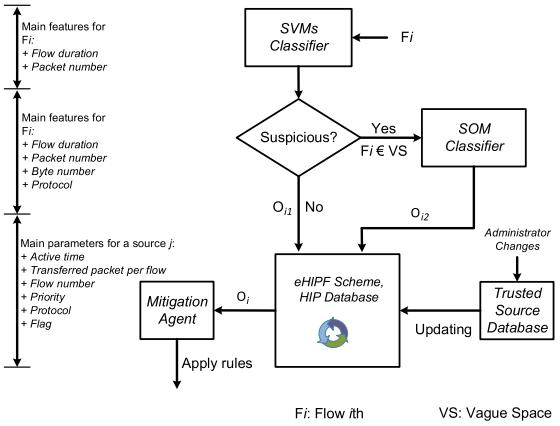

FIGURE 5. System Process Logic.

2) MACHINE LEARNING TRAINING AND CLASSIFYING

Once the proposed scheme is activated, and machine learning training processes will take place at the initialization stage. In this process, *SVM-k* and *SOM* classifiers are trained by the appropriate ready-made datasets from the Training Database. Note that, in this scheme, the *Training Database* is continuously updated using the flow attributes collected from the above loop. In a preset time, which may be defined by a network administrator, the *SVM-k* and the *SOM* will be replaced by a *SVM-k* and a *SOM* that are trained by using the updated database because the *SVM* and *SOM* are not an online learning algorithms [39]. By doing so, the proposed mechanism can adapt well to various network states.

We already utilized the enhanced workable *SVMs-SOM* combination in [9]. Therefore, we briefly summarize our proposed mechanism as follows

- Each *SVM-k* classifies an input vector (a tuple of flow's attributes) basing on the distance d from their margins to the hyperplane, and a decision will be given based on the input's position that is pointed out in the *SVM-k* representation.
- If $d < \frac{1}{\|w\|}$ where $\vec{w}$ is the normal vector to the hyperplane, the input vector is believed as a suspicious pattern and located in a vague space (VS) because *SVM-k* cannot find any groups (normal or abnormal) for the considered input vector. Then, the input information will be forwarded to the Self-Organizing Map to ask for a high-accurate decision. Otherwise, the *SVM-k* produces and forwards an output to the *eHIPF* for next processes.
- After receiving the input from one of *SVM-k*, the *SOM* map, which is constructed by a training process using a ready-made dataset and makes decision based on the weight computation of the neurons, it will compute the distance from the input vector to every node or neuron in it's map to choose the Best Matching Unit. Next, the SOM gives out and forwards an output to the *eHIPF* for next processes.

3) SYSTEM PROCESS LOGIC

To represent our introduced scheme, the system control-flow logic is shown in Figure 5. At first, an input F*i* or an attribute tuple of flow F*i*, which is previously processed by the Raw Data Processing module, is fed into one of the *SVM-k*. Then, if the F*i*'s position locates outside the vague space (F$i \in G_1$ or F$i \in G_2$), an output O_{i1} will be sent immediately to the *eHIPF* module. Otherwise, F$i \in VS$, the F*i*'s information is forwarded to the next machine learning classifier (*SOM* map). The *SOM* takes more F*i*'s attributes in order to make a more accurate prediction by using the *SOM* neurons, and then it produces an output O_{i1} for F*i* and also forwards to the *eHIPF*. The *eHIPF* scheme is responsible for deeply attack checking based on the traffic history of the incoming traffic pattern F*i* (discussed in IV-C) and produces an output O_{i2}. After gathering outputs O_{i1} and O_{i2}, final decision is made by following rules:

- If O_{i1} is ATTACK and O_{i2} is ATTACK, O_i is ATTACK;
- If O_{i1} is ATTACK and O_{i2} is NORMAL, O_i is NORMAL (Consider at next observation);
- If O_{i1} is NORMAL and O_{i2} is ATTACK, O_i is NORMAL (Consider at next observation);
- If O_{i1} is NORMAL and O_{i2} is NORMAL, O_i is NORMAL.

By doing so, the F*i* will be grouped and classified based on a various adjustment aspects. The last procedure is at the Mitigation Agent module that formulates policies for specific types of attacks and sends rules to OpenFlow switches with

the purpose of reducing attacks. Regarding normal traffic flows, we do nothing to keep them alive in the flow-tables. These mentioned processes are summarized in Algorithm 1.

Algorithm 1 Proposed DDoS Defense Scheme

$N \leftarrow$ A set of flows
$F_i \leftarrow$ Attribute tuple of flow i
$O_i \leftarrow$ Final decision of F_i
$O_{i1} \leftarrow$ Output of $SVM - k$ or SOM classifier
$O_{i2} \leftarrow$ Output of $eHIPF$
$r_i \leftarrow$ A generated rule for an attack flow F_i
$R \leftarrow$ A set of rules in each loop
loop
  for $i = 1$ to N **do**
    Feed F_i into SVM-k
    if $F_i \in VS$ **then**
      Suspicious Flow
      Pass $F_i \rightarrow SOM \Rightarrow O_{i1}$
    else
      Produce O_{i1}
    end if
    Forward $F_i \rightarrow eHIPF \Rightarrow O_{i2}$
    Process (O_{i1},O_{i2}) $\rightarrow O_i$
    if O_i is ATTACK **then**
      Forward (F_i,O_i) $\rightarrow$ *Mitigation Agent* $\Rightarrow r_i$
      R $\leftarrow$ (R + r_i)
    else
      continue
    end if
  end for
end loop
return: R

C. ENHANCED HISTORY-BASED IP FILTERING SCHEME AND MITIGATION AGENT

1) HISTORY-BASED IP DATABASE

To begin with, we first define a set of parameters (or rules) of a source IP_j ($y \leq j \leq x$) which is used to differentiate normal sources from their illegal counterparts.

The first parameter is the active time of a source IP_j denoted as at_j ($at_j \leq h$). This parameter not only denotes how long time a source has been connecting to a service on the cloud, but also proves that the source IP_j has not sent any attack traffic to our system under the view of a cloud provider. Hence, at_j is considered to be a key parameter in making accurate decisions on attack detection (the output O_{i2} in Algorithm 1). As mentioned in section IV-A, the value of at_j of the source IP_j will be reset to 0 after h hours, and it will then recount up to h hours. The second parameter is the number of flows of a source IP_j, n_{con_j}, which shows how many incoming flows have been established to a service on the cloud, where $n_{con_j} \geq 1$. It is reasonable to consider only request flows because the NFV and SFC technologies, in which the return traffic is always another route for a better service response or for other purposes. The ICMP protocol is known as a supporting protocol in the Internet protocol suite, and it is just used to check whether a requested service is not available or a host or router could not be reached. Thus, a normal source IP_j usually sends few ICMP request packets to a destination, and each ICMP (generally attack *Type I* mentioned in Section III-B) source IP_j only generates one flow in flow-tables of a SDN switch. In contrast, a TCP or UDP (generally attack *Type II* traffic mentioned in Section III-B) source IP_j can generate one or more flows in a SDN switch. From the above analyses, we define the third parameter, Pf_j (the average number of packets per flow of the source IP_j) to differentiate an abnormal source IP_j from their normal counterparts as follows

$$\begin{cases} Pf_j = t_{pkt_j}, & Attack\ Type\ I, \\ Pf_j = \dfrac{t_{pkt_j}}{n_{con_j}}, & Attack\ Type\ II, \end{cases} \tag{2}$$

where n_{con_j} is the flow number of the source IP_j and t_{pkt_j} is the transferred packets of the source IP_j to the cloud. The traffic protocol of the source IP_j is a major point to classify traffic type into two main attack types as described in III-B, Pro_j. For example, the ICMP flooding attack belongs to the *Type I* ($Pro_j = 1$), while TCP SYN flooding is classified as *Type II* ($Pro_j = 2$). The next judged parameter is the priority of a source IP_j denoted as Pri_j that distinguishes among trusted ($Pri_j = 1$), normal ($Pri_j = 2$) and unidentified ($Pri_j = 3$) sources. The last parameter we introduce is the flag, $Flag_j$, which shows the status of the source IP_j. We have two statuses for a source IP_j: *attack* and *normal* which represent $Flag_j = -1, 1$, respectively. A new source is assigned as a normal source at the beginning.

Note that at each observation, the Update Agent collects values of at_j, n_{con_j}, Pf_j, Pro_j and Pri_j of each active source IP_j and updates to a database. From the above definitions, the tuple of parameters of a source IP_j is formed as $IP_j = (at_j, n_{con_j}, Pf_j, Pro_j, Pri_j\ Flag_j)$ and is an entry in the database. For each search process, the search engine relies on two matches: IP_j and Pro_j to distinguish the IP sources.

2) ENHANCED HISTORY-BASED IP FILTERING SCHEME

As shown in [20]–[22], the experiments are conducted by analyzing some datasets and finding some hard thresholds to distinguish the attack sources from normal sources. For example, [20] recommend that a TCP traffic the number of packets of a normal source IP_j is not less than 3 packets for a normal TCP session. Because these previous researches set hard threshold and datasets to determine the attack source, they cannot be adaptive to changes in the real time traffic characteristics. Furthermore, the use of a long-day dataset is not an efficient way to make the system be fresh after a long period of operation. Hence, in this paper, we introduce an enhanced scheme to detect abnormal sources under DDoS attacks based on the history of network traffic which is referred *eHIPF* scheme.

We now investigate how *eHIPF* scheme efficiently classifies normal and attack sources using a History-based IP

Database (HIP Database) described in IV-C.1 at each observation using the following phases:

Phase 1 (Threshold Initialization): At the beginning, we generate a specific set of thresholds $Ini = \{(at_{ini}, w_1), (Pf_{ini}, w_2), (n_{con_{ini}}, w_3)\}$ for each type of traffic or protocol, where w_1, w_2, w_3 are weights of *Ini* attributes that indicate the importance of a specific attribute among others. To choose appropriate values for these weights, from our deep discussion in Section III-B. Normally, each specific attack has one or two prominent characteristics (e.g. ICMP flooding has a large number of packets in one request flow). Hence, we clearly know which attributes have the highest effects on detecting a DDoS source. Therefore, we set w_1, w_2, w_3 values according to the impact of its attributes on the detection decision. In order to initiate the *Ini* sets, we utilize data from the Training Database of the machine learning classifiers because we also use this database to train for *SVM* and *SOM* algorithms. However, this is just an initialization step, the *Ini* sets will no longer be used if new sources are added to the HIP Database.

Phase 2 (Real-Time System Processing): When the system is on running-time, based on the collected statistics at each observation, we first extract and update all active source IP addresses by *protocol* to the HIP Database using the Update Agent. For some beginning observations, we use the *Ini* sets to classify normal and abnormal sources for each type of traffic protocol and the *Ini* sets will be maintained separately until the proposed system detects any of the attack sources. For the observation t, the HIP Database may add some new sources. Then, these sources are verified to find their *Flag* using ready-made *Ini* sets. The value of the corresponding *Ini* set will replaced for the next observation $(t+1)$ according to our proposed Algorithm 2.

We provide an example of the source classification using our presented *eHIPF* scheme. Regarding to the ICMP DDoS attack (*Type I*), we assume that $B_{(t+1)_{ICMP}} = \{(15.56, 0.3), (50.5, 0.5), (1.0, 0.2)\}$ is already calculated in t observation, and we get 1000 active IP sources at $(t+1)$ observation. Next, we compare these source's attributes with $B_{(t+1)_{ICMP}}$ to classify the ICMP attack and normal sources according to the Algorithm 3. For instance, a new ICMP source $A.B.C.D = \{3.6, 1.0, 67.0, 1.0, 1.0, 1\}$. According to the Algorithm 3 we can have: $X_{ICMP} = 30.118 < X_{A.B.C.D} = 34.78$. Therefore, we can conclude that the source $A.B.C.D$ is an ICMP flooding attack source.

To prove the proposed *eHIPF* scheme as an enhancement of the original *HIPF* solutions, we conducted experiments and comparisons in Section V.

3) MITIGATION AGENT

To meet our two goals discussed in Section IV-A: The proposed scheme has to detect the attack sources as fast as possible, and to prevent abnormal traffic from accessing VNFs inside the cloud as much as possible. Accordingly, we apply various protection techniques for different attack types or protocols.

Algorithm 2 Boundary Calculation for Next (t+1) Observation

$z \leftarrow$ The number of all IP sources in the system
$B_{(t+1)} \leftarrow$ Boundary set at next (t+1) observation
$w_1, w_2, w_3 \leftarrow$ Attribute weights of the boundary set
Initialize $at = Pf = n_{con} = 0.0$
$B_{(t+1)} = \{(at, w_1), (Pf, w_2), (n_{con}, w_3)\}$
$c \leftarrow 0$
for $j = 1$ to z **do**
  if $Flag_j = -1$ (Consider attack sources) **then**
    $c+ = 1$
    $at+ = at_j$
    $Pf+ = Pf_j$
    $n_{con}+ = n_{conj}$
  end if
end for
$B_{(t+1)} = \left\{(\frac{at}{c}, w_1), (\frac{Pf}{c}, w_2), (\frac{n_{con}}{c}, w_3)\right\}$
return: $B_{(t+1)}$

Algorithm 3 *eHIPF* Abnormal Source Detection

$B_{(t+1)_i} = \{(at_i, w_{1_i}), (Pf_i, w_{2_i}), (n_{con_i}, w_{3_i})\} \leftarrow$ the boundary set of the protocol i given at the tth observation
$(at_j, n_{con_j}, Pf_j, Pro_j, Pri_j, Flag_j) \leftarrow$ A set of attributes of a source IP_j that is collected at the $(t+1)$ observation
$X_i = at_i * w_{1_i} + Pf_i * w_{2_i} + n_{con_i} * w_{3_i}$
$X_j = at_j * w_{1_i} + Pf_j * w_{2_i} + n_{con_j} * w_{3_i}$
if $X_i \leq X_j$ **then**
  $Flag_j = -1$ {Attack source}
else
  $Flag_j = 1$ {Normal source}
end if
return: $Flag_j$

Regarding attack *Type I*, in which attack sources intend to create one or two flows from a client. They always send a large number of packets, for example ICMP Flooding. To prevent this attack, the Mitigation Agent sends a *flow mod* message [36] with a *drop* action and a preset $hard - timeout$ value to the edge OpenFlow switch. This means that every packet of the flooding source will be dropped at the border switch and cannot reach inside the cloud. This action is applied as soon as the *eHIPF* scheme produces an attack output to the Mitigation Agent. Note that these policy-applied flows will no longer be considered as an input at next observation. By doing so, the proposed mechanism can reduce the computational resources of the cloud control plane in the SDN-based architecture.

Attack *Type II* generates a massive number of flows to the victim with a small number of packets, for instance with a TCP SYN DDoS attack. To tackle this attack, the Mitigation Agent sends a *flow_mod* message with a *delete* action to the edge OpenFlow switch and informs the forwarding engine of the SDN controller to drop *packet_in* messages

of attacking sources which require for new flow installation at the OpenFlow switch. This policy removes all abnormal flows and prevents new attack flows in case an DDoS attack is happening. Hence, it makes VNFs not getting overloaded and ensures the smooth cloud operation. In addition, at the next observation the number of collected flows will significantly decrease due to the attack flow deletion from the switch. As a result, this policy can save as much computational resources of the cloud control plane as possible.

V. EXPERIMENTS

First, this section gives readers a demonstration of the *eHIPF* scheme compared to the *HIPF* [20], [21] in terms of the detection speed. Next, we present an elaborate implementation and a comprehensive result analysis of the proposed scheme.

A. EHIPF EXPERIMENT

With the purpose of the *eHIPF* scheme deployment, we set up a simple topology including a SDN controller, an OpenFlow switch (HP E3800 [40]), an attacker host, a Web server and a connection to our laboratory network. In this setup, the SDN controller connects to the OpenFlow switch via a secure connection, and two hosts, and the Laboratory network is assigned to three switch ports. We coded and placed *eHIPF* and *HIPF* [20], [21] modules in the SDN controller respectively, and run these as applications. We perform two separate scheme experiments by activating the modules, both accessing from the laboratory network and generating DDoS attack traffic from the attacker host installed BoNeSi (DDoS attack tool) [41] to the Web server machine.

Before running the system, we extract and take $Ini = \{(at_{ini}, w_1), (Pf_{ini}, w_2), (n_{con_{ini}}, w_3)\}$ sets for each protocol from the CAIDA datasets [42], [43]. In addition, we use a pool of 900 fake source IP addresses in the attacker host and a set of 100 trusted sources in the laboratory hosts to simultaneously generate attack and normal traffic. However, to test the speed of detection of both approaches, the attack tool begins with different attack sources corresponding to various arrival rates $\lambda_1 = 100$, $\lambda_2 = 300$, $\lambda_3 = 500$. We tested two typical attacks ICMP flooding (*Type I*) and TCP SYN flooding (*Type II*) with the observation time, $t_{ob} = 3$ seconds. The results of the experiment are detailed in Section VI.

B. SDN-BASED CLOUD IMPLEMENTATION AND TEST PREPARATION

We implement our proposed solution and compare to other machine learning-based solutions: *SVMs-SOM* [9], *SVMs* [25] and *SOM* [24]. Figure 6 shows our testing topology which consists of a SDN controller, OpenStack platform (a Controller-Network node and three Compute nodes running OvS drivers for cloud networking), an OpenFlow switch HP E3800, a Router, the laboratory's LAN network, the Internet connection and botnets. In order to emulate real DDoS attack scenarios, three servers installed the BoNeSi DDoS attack tool are used to generate different attack volumes and

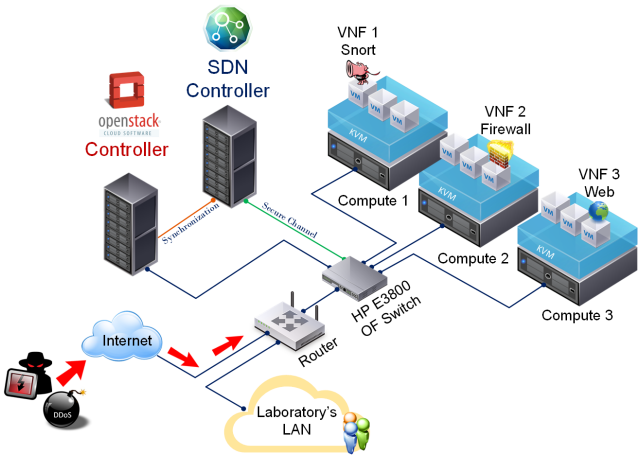

FIGURE 6. Experimental topology.

rates to the victim. With respect to the service function chaining setup, we configure flows for a service chain including three applications on three OpenStack Compute nodes: VNF 01 - *Snort*, VNF 02 - *Firewall* and VNF 03 - *Web server*. In addition, a synchronized connection is established between the SDN and the OpenStack controllers for the cloud information and adjustments that are needed.

Machine learning training experiments are conducted using both normal and attack datasets from CAIDA Datasets [42], [43] (Normal traffic - on 21*st* May, 2015 and DDoS attack traffic - on 04*th* August, 2007) to train *SVM* and a *SOM* classifiers. CAIDA is one of the most credible datasets which collects diverse real network traffic types, including Web, FTP, Ping and etc at different locations worldwide. In a legitimate dataset, the TCP protocol packet occupies a massive 89%, a merely 6% is ICMP packets and other protocols make up only 5%. Whereas, in the abnormal dataset, a small proportion of just under 6% are TCP packets and ICMP packets becomes a main protocol used for DoS attacks with a vast majority 93%, and there is merely 1% corresponding to other protocols. From the two aforementioned datasets, both TCP and ICMP protocols account for the highest proportion. Thus, in this work, we focus on DoS attacks using TCP and ICMP protocols.

To generate real DDoS attack traffic to a predefined SFC, the DDoS attack tool named BoNeSi [41] is installed in three bots to generate abnormal flows, while normal flows are made from the laboratory's network which is considered as a trusted subnet or every source IP from this network is believed to be a normal and trusted source. We set the value of the key parameters to test our introduced mechanism, as shown in Table 2.

C. TESTING CONDUCTION

The testing procedure is divided into three main periods for each mechanism and can be summarized as follows:

TABLE 2. Key testing parameters.

Feature	Value
Pool of trusted sources	$y = 100$
Pool of attack sources	$z - x = 400$
Attack arrival rate	$\lambda = 500$
Weights of Boundary	w_1=w_2=w_3=1.0
Observation time (second)	$t_{ob} = 5$
DDoS attacks time (second)	$t_{attack} = 100$
HIP Database history time (hours)	$h = 12$
Number of *SVM* classifiers	2
Number *SOM* neurons	40x40

- At the beginning of the experiment, we generate traffic from the trusted sources and BoNeSi tool using only the ICMP protocol.
- Next, we use the same configuration to send only TCP traffic.
- Finally, we attack the service chain using both ICMP and TCP traffic protocols and each protocol has a same number of fake sources.

VI. RESULT ANALYSIS

This section demonstrates detailed experiments along with comprehensive analyses to evaluate our proposed mechanism compared to other solutions.

A. EHIPF ENHANCEMENT COMPARISON

As mentioned in Section V-A, we carried out experiments for two different solutions, *eHIPF* and *HIPF* [20] under DDoS attacks to evaluate an enhancement in our proposed scheme in terms of the speed of detection. Figure 7 illustrates the number of abnormal sources detected by time for two flooding attack types (ICMP and TCP SYN). It is evident that both *eHIPF* schemes totally outperform the normal *HIPF* using hard thresholds to detect attack sources. First, we generate three different arrival rate levels of attack source groups ($\lambda_1 = 100$, $\lambda_2 = 300$, $\lambda_3 = 500$), then it would result in various incoming traffic to the victim. The normal *HIPF* based on a hard threshold to make decision, has to wait for some parameters being over the threshold. Meanwhile, *eHIPF* adapts its self to the volume of traffic, and varies the threshold according to the Algorithm 2. Hence, the result is understandable in that the *eHIPF* always has higher detected sources at the same time with normal *HIPF*. In other words, the *eHIPF* enhances much the speed of detection when compared to the traditional solution.

B. DETECTION RATE, ACCURACY AND FALSE ALARM RATE ENHANCEMENTS

Figure 8 shows that regarding Detection rate and Accuracy, our novel approach always accounts for the highest rate, 99.27% and 99.30% respectively, which is slightly higher than the *SVMs-SOM* solution with 98.47% and 97.62%, and clearly outperforms the original *SVMs* and *SOM* algorithms. With respect to False alarm rate, the proposed scheme dominates the production of wrong warnings when only occupies 0.67%, while the rate of incorrect alarm generated by the *SVMs-SOM* is above 3.20%, *SOM* and *SVMs* are around 6.40%. To achieve these improvements, the main reason is that the *eHIPF* gives out sensible advice to decide which groups flows belong to (attack or normal group). For example, a normal TCP session sends very first packets to a destination. Due to the short observation time, a collected TCP flow would look like a TCP SYN flooding attack flow with 1 or 2 packets. Consequently, the system without *eHIPF* will make a wrong decision. This mistake leads to many thorny problems in the cloud, such as a packet_in process or service delay. In conclusion, our new mechanism is more efficient than other methods for all evaluation criteria.

C. TRAFFIC FLOW OCCUPATION AT DATA PLANE

As demonstrated in Figure 9, it is clear that there are major differences among the four tested mechanisms. In the first stage, attackers send only ICMP flooding traffic, and the number of flow occupations in the switch flow-tables are the same because each source IP is able to open only one request

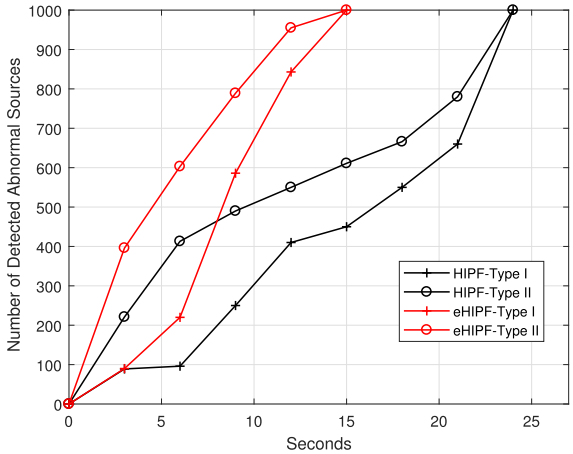

FIGURE 7. Detection performance of *eHIPF* and *HIPF*.

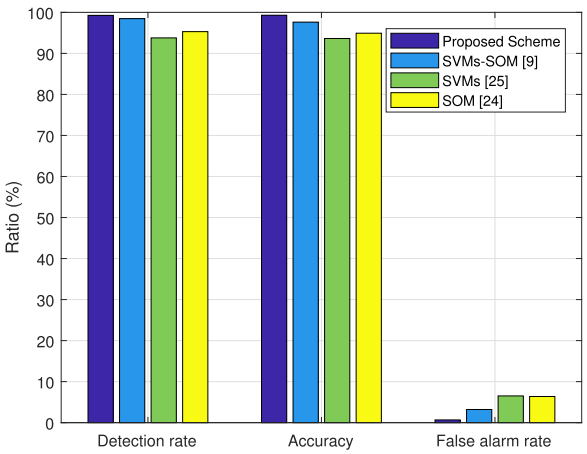

FIGURE 8. Detection rate, Accuracy and False alarm rate comparison.

TABLE 3. Bandwidth occupation in control-data plane secure channel.

Type	Proposed Scheme (Kbps)	*SVMs – SOM* [9] (Kbps)	*SVMs* [25] (Kbps)	*SOM* [24] (Kbps)
Type I : ICMP Flooding Attack	705	734	745	762
Type II : TCP SYN Flooding Attack	1,367	1,378	1,401	1,380
Both Types: ICMP and TCP SYN Flooding Attacks	1,004	1,110	1,154	1,177

flow to the destination. However, we apply the *drop* action for attack flows. Therefore, the number of flow is equal to both the number of abnormal and trusted sources. In case, the attack tool only generates TCP SYN attack traffic, we can see the proposed solution and the *SVMs-SOM* outperform two original schemes, in which the proposed scheme is the best flow saver with approximately 3000 flows in the switch all time. This lower flow occupation of our novel mechanism can be explained by successfully applying the *eHIPF* scheme to produce fewer wrong decisions, and this leads to an abnormal detection rate that is higher than that of other cases, and more attack flows are removed from the switch as soon as the final output from the *eHIPF* scheme is obtained.

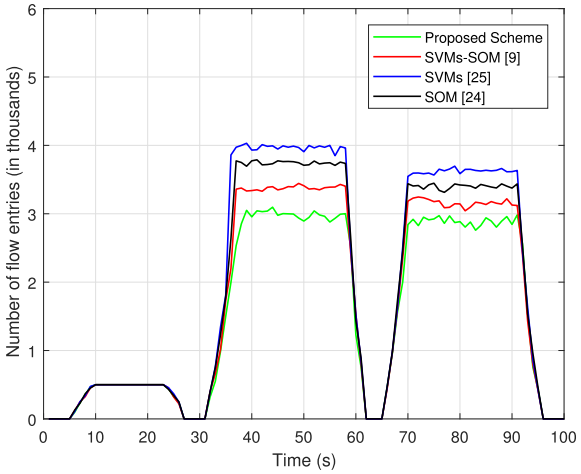

FIGURE 9. Flow rule occupation in flow-tables of the HP 3800 switch.

D. CONTROL PLANE RESOURCE CONSUMPTION

1) SDN CONTROLLER'S CPU UTILIZATION

The SDN controller's CPU consumption is considered to be the main criteria to assess the proposed solution performance when compared to other schemes. First, Figure 10 indicates the use of the SDN controller's CPU of three testing scenarios. Overall, it is easy to realize a better performance by our proposed method. In the ICMP traffic period, the *SVMs* scheme is the least computational resources because of the light *SVM* algorithm and the same flow number level, followed by our novel solution just under 30%. However, during the TCP traffic period, our proposed scheme is the best one in spite of the lowest flow number and *packet_in* number in each observation, the *SOM* (58.5% on average) and *SVMs* (42% on average) are the worst cases with the same reason for the flow quantity in one observation. The last is the combined

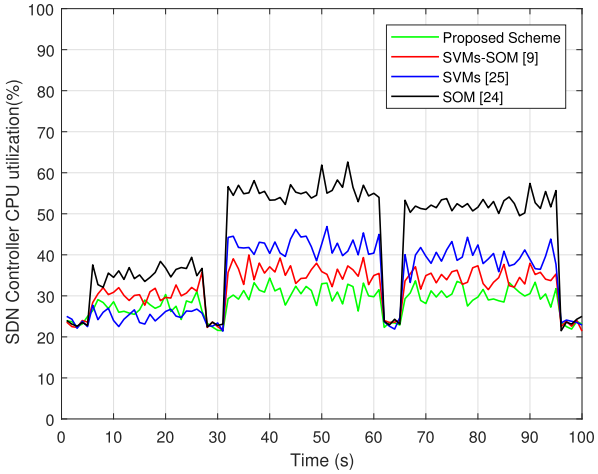

FIGURE 10. SDN controller's CPU utilization comparison.

period, and the results are the same as those in the second scenario. It is clear that our proposed solution always keeps the SDN controller stable at around 30% in consuming its computational resources.

2) *PACKET_IN* RATE AND BANDWIDTH OCCUPATION IN CONTROL-DATA PLANE CHANNEL

We collect the *packet_in* rate to the SDN controller demonstrated in Figure 11 and the bandwidth occupation in Control-Data plane channel is shown in Table 3. At a glance, there are no significant differences between the solutions in both measurements. Regarding the *packet_in* rate, in all three

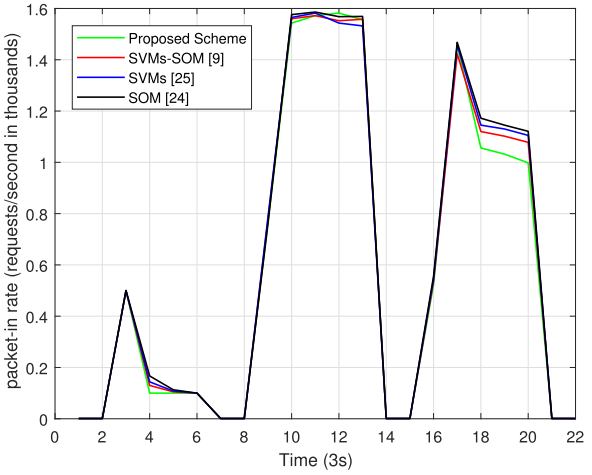

FIGURE 11. Number of *packet_in* requests arriving SDN Controller.

attack periods, the proposed scheme and others show a same trend at each period. This is understandable because the incoming ICMP traffic opens one flow for one source while TCP traffic generates random *Layer* 4 port number. Hence, after being detected the ICMP traffic will no longer send *packet_in* messages to the SDN controller. However, the TCP traffic still does. Regarding the bandwidth occupation in the channel, our proposed scheme is slightly less than others due to the fact that more precise decision making does not result in more control messages from the SDN controller.

E. SERVICE CHAIN OPERATION

Previous assessments mainly focus on the OpenFlow switch and the SDN controller as well as the secure channel. Now, we evaluate how DDoS attacks affect the service chain on the SDN-based cloud and which problems can be solved by the proposed mechanism when compared with other solutions.

1) VNF 01 - SNORT CPU UTILIZATION

From the demonstration in Figure 12, we can see that our proposed solution always accounts for the smallest percentage of the *Snort* CPU usage. This enhancement is quiet reasonable because the proposed scheme offers faster detection, more accurate decision making and is the right policy maker. These things keep the service chain traffic as low as possible for the *Snort* process. Hence, it significantly reduces the usage of the *Snort* CPU. Regarding the trend of the CPU usage, in the ICMP traffic case, although at the early stage the *Snort* is under high pressure from the high traffic volume, the CPU usage then goes down steadily because the attack sources are detected and the Mitigation Agent applies rules to drop all incoming packets. Therefore, *Snort* can save much of the processing resources. Meanwhile, the second case witnesses different stable uses of the *Snort* CPU for all solutions. This is mainly based on the quantity of flows and traffic in each flow (normally 1 or 2 packets per request flow. Hence, the *Snort* resource consumption adapts to the flow number of each mechanism. Finally, in the last case, it is the combined attack traffic with an equal number of attack sources. The trend still increases at the beginning of the attack, and it then goes down and stands at the same level until the attack finishes.

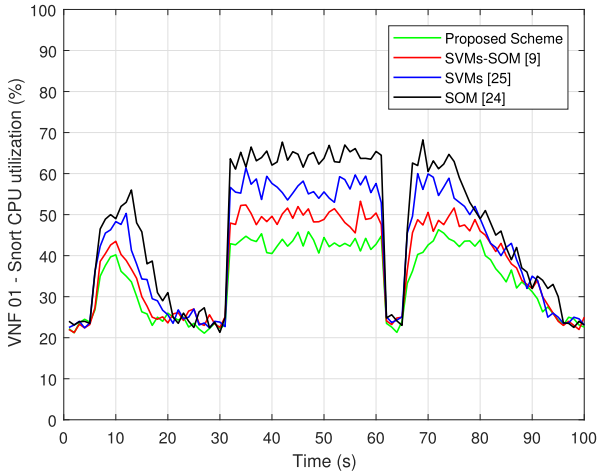

FIGURE 12. VNF 01 - Snort CPU utilization comparison.

2) QUALITY OF SERVICE

To assess the effects of DDoS attacks on the service chain customer with the four mentioned solutions, we observe and score the Quality of Service (QoS) using two criteria: Request delay time and Packet loss rate.

3) TRAFFIC DELAY

As demonstrated in Figure 13, this provides readers a request delay of a service chain on the SDN-based cloud environment under different DDoS attack levels for four mechanisms. To observe this experiment, we simply generate a *Ping* command to the VNF 03 - Web server from a trusted host in the laboratory network, and we use the tool named Wireshark [44] to measure the service chain traffic delay. It is evident that the fastest response from the Web server is provided by the proposed scheme from just under 60 (ms) to around 120 (ms) on average, which is followed by the *SVMs-SOM*, *SVMs* and *SOM*, respectively. Once again, our novel proposal presents a better judgment among others in terms of the service chain traffic delay.

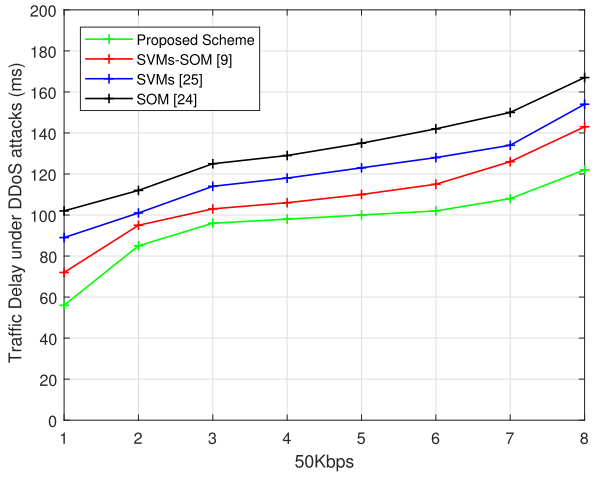

FIGURE 13. Comparison in service chain traffic delay.

4) LEGAL USER'S PACKET LOSS RATE

The next criterion is chosen to evaluate the proposed scheme is the packet loss rate of a legitimate user to a service chain. To measure this experiment, we also do with 100 *ICMP* ping packets and the same technique under various DDoS attack levels for four testing solutions. Due to the traffic congestion that causes the packet to drop under saturation attacks, this makes the packet loss rate increase when the attack level goes up as shown in Figure 14. As we can see, that our proposed

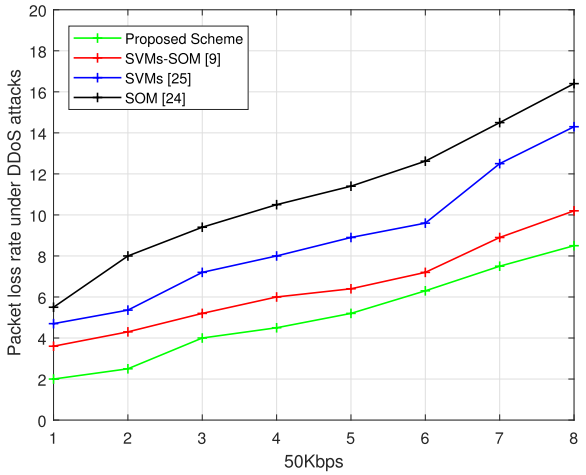

FIGURE 14. **Comparison in packet loss rate.**

still achieves the best performance in all testing attack degrees while others cannot.

VII. CONCLUSION AND FUTURE WORK

In this paper, we propose a novel mechanism to handle DDoS attacks in SDN-based cloud environment. This proposal not only protects cloud infrastructure from being overloaded resulted by DDoS attacks in both control and data planes, but also brings a better quality of service to cloud customers. We present a new machine learning hybrid model for classification based on utilizing the advantages of Support Vector Machine and Self Organizing Map algorithms. We also propose an enhanced History-based IP Filtering scheme to improve the detection performance in recognizing DDoS attack source IP addresses. Finally, we introduce a novel DDoS attack defender which is based on both machine learning and history-based techniques. Through experimental results conducted in various DDoS attacks levels, we prove that the novel mechanism can be an effective and innovative approach to face DDoS attacks in SDN-based cloud environment.

As our future work, we expect to design some new functional modules to enhance the *packet_in* process in the proposed mechanism with the purpose of recognizing malicious *packet_in* messages from the data plane and optimizing the bandwidth occupation in the secure communication channel. In addition, we plan to compare the proposed scheme to other machine learning techniques (e.g., deep learning) using more evaluation criteria.

REFERENCES

[1] B. A. A. Nunes *et al.*, "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 3, pp. 1617–1634, 3rd Quart., 2014.

[2] *Network Functions Virtualisation: Introductory White Paper*. [Online]. Available: https://portal.etsi.org/nfv/nfv-white-paper2.pdf

[3] *Service Function Chaining*. [Online]. Available: https://datatracker.ietf.org/wg/sfc/documents/

[4] *Welcome to OpenStack Documentation*. [Online]. Available: http://docs.openstack.org/

[5] G. Somani, M. S. Gaur, D. Sanghi, M. Conti, and R. Buyya, "DDoS attacks in cloud computing: Issues, taxonomy, and future directions," *ACM Comput. Surv.* [Online]. Available: https://arxiv.org/abs/1512.08187

[6] K. Lab, "DDoS attacks in Q1 2018," Tech. Rep., 2018.

[7] M. D. Firoozjaei, J. P. Jeong, H. Ko, and H. Kim, "Security challenges with network functions virtualization," *Future Gener. Comput. Syst.*, vol. 67, pp. 315–324, Feb. 2017.

[8] *SDN Security Considerations in the Data Center*. [Online]. Available: https://www.opennetworking.org/images/stories/downloads/sdn-resources/solution-briefs/sb-security-data-center.pdf

[9] T. V. Phan, N. K. Bao, and M. Park, "A novel hybrid flow-based handler with DDoS attacks in software-defined networking," in *Proc. 13th IEEE Int. Conf. Adv. Trusted Comput.*, Toulouse, France, Jul. 2016, pp. 350–357.

[10] J. Shawe-Taylor and N. Cristianini, *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge, U.K.: Cambridge Univ. Press, 2000.

[11] T. Kohonen, *Self-Organizing Maps*. Secaucus, NJ, USA: Springer-Verlag, 1997.

[12] J. Laaksonen and T. Honkela, "Advances in self-organizing maps," in *Proc. 8th Int. Workshop WSOM*, Helsinki, Finland, Jun. 2011.

[13] Q. Yan, F. R. Yu, Q. Gong, and J. Li, "Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: A survey, some research issues, and challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 602–622, 1st Quart., 2016.

[14] I. Ahmad, S. Namal, M. Ylianttila, and A. Gurtov, "Security in software defined networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2317–2346, 4th Quart., 2015.

[15] H. Wang, L. Xu, and G. Gu, "FloodGuard: A DoS attack prevention extension in software-defined networks," in *Proc. 45th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Jun. 2015, pp. 239–250.

[16] P. Van Trung, T. T. Huong, D. Van Tuyen, D. M. Duc, N. H. Thanh, and A. Marshall, "A multi-criteria-based DDoS-attack prevention solution using software defined networking," in *Proc. IEEE Int. Conf. Adv. Technol. Commun.*, Oct. 2015, pp. 308–313.

[17] K. Giotis, C. Argyropoulos, G. Androulidakis, D. Kalogeras, and V. Maglaris, "Combining OpenFlow and sFlow for an effective and scalable anomaly detection and mitigation mechanism on SDN environments," *Comput. Netw.*, vol. 62, no. 5, pp. 122–136, 2014.

[18] S. A. Mehdi, J. Khalid, and S. A. Khayam, "Revisiting traffic anomaly detection using software defined networking," in *Recent Advances in Intrusion Detection*, R. Sommer, D. Balzarotti, and G. Maier, Eds. 2011.

[19] S. K. Fayaz, Y. Tobioka, and V. Sekar, "Bohatei: Flexible and elastic DDoS defense," in *Proc. 24th USENIX Secur. Symp.*, Washington, DC, USA, 2015, pp. 817–832.

[20] T. Peng, C. Leckie, and K. Ramamohanarao, "Protection from distributed denial of service attacks using history-based IP filtering," in *Proc. 3rd IEEE Int. Conf. Commun. (ICC)*, May 2003, pp. 482–486.

[21] N.-N. Dao, J. Kim, M. Park, and S. Cho, "Adaptive suspicious prevention for defending DoS attacks in SDN-based convergent networks," *PLoS ONE*, vol. 11, no. 8, Aug. 2016, Art. no. e0160375.

[22] M. Goldstein, C. Lampert, M. Reif, A. Stahl, and T. Breuel, "Bayes optimal DDoS mitigation by adaptive history-based IP filtering," in *Proc. 7th Int. Conf. Netw.*, Apr. 2008, pp. 174–179.

[23] J. N. Bakker, B. Ng, and W. K. G. Seah, "Can machine learning techniques be effectively used in real networks against DDoS attacks?" in *Proc. 27th Int. Conf. Comput. Commun. Netw.*, Hangzhou, China, Jul./Aug. 2018, pp. 1–6.

[24] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *Proc. 35th IEEE Conf. Local Comput. Netw.*, Denver, CO, USA, Oct. 2010, pp. 408–415.

[25] T. V. Phan, T. Van Toan, D. Van Tuyen, T. T. Huong, and N. H. Thanh, "OpenFlowSIA: An optimized protection scheme for software-defined networks from flooding attacks," in *Proc. IEEE 6th Int. Conf. Commun. Electron. (ICCE)*, Jul. 2016, pp. 13–18.

[26] T. V. Phan, N. K. Bao, and M. Park, "Distributed-SOM: A novel performance bottleneck handler for large-sized software-defined networks under flooding attacks," *J. Netw. Comput. Appl.*, vol. 91, pp. 14–25, Aug. 2017.

[27] Q. Niyaz, W. Sun, and A. Y. Javaid, "A deep learning based DDoS detection system in software-defined networking (SDN)," in *Proc. EAI Endorsed Trans. Secur. Saf.*, vol. 4, 2017.

[28] N. Meti, D. G. Narayan, and V. P. Baligar, "Detection of distributed denial of service attacks using machine learning algorithms in software defined networks," in *Proc. Int. Conf. Adv. Comput., Commun. Inform. (ICACCI)*, Udupi, India, Sep. 2017, pp. 1366–1371.

[29] Y. Li and M. Chen, "Software-defined network function virtualization: A survey," *IEEE Access*, vol. 3, pp. 2542–2553, Dec. 2015.
[30] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," in *Proc. USENIX Workshop Hot Topics Manage. Internet, Cloud, Enterprise Netw. Services (Hot-ICE)*, vol. 54, 2012, pp. 1–6.
[31] M. Monaco, O. Michel, and E. Keller, "Applying operating system principles to SDN controller design," in *Proc. 12th ACM Workshop Hot Topics Netw.*, 2013, Art. no. 2.
[32] A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, and R. Kompella, "Towards an elastic distributed SDN controller," in *Proc. 2nd ACM SIGCOMM Workshop Hot Topics Softw. Defined Netw.*, 2013, pp. 7–12.
[33] N. Gude *et al.*, "NOX: Towards an operating system for networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 3, pp. 105–110, 2008.
[34] C. Douligeris and A. Mitrokotsa, "DDoS attacks and defense mechanisms: Classification and state-of-the-art," *Comput. Netw.*, vol. 44, pp. 643–666, Apr. 2004.
[35] A. T. Pascoal, Y. G. Dantas, I. E. Fonseca, and V. Nigam, "Slow TCAM Exhaustion DDoS Attack," in *Proc. 32nd Int. Conf. ICT Syst. Secur. Privacy Protection (IFIP SEC)*, 2017, pp. 17–31.
[36] Message Layer Definition. *OpenFlow Messages*. Accessed: 2018. [Online]. Available: http://flowgrammable.org/sdn/openflow/message-layer
[37] M. Zukerman, "Introduction to queueing theory stochastic teletraffic models," Tech. Rep., 2016.
[38] T. Kim, S. Kim, K. Lee, and S. Park, "A QoS assured network service chaining algorithm in network function virtualization architecture," in *Proc. 15th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, May 2015, pp. 1221–1224.
[39] S. Shalev-Shwartz, *Online Learning: Theory, Algorithms, and Applications*. 2007.
[40] *HP 3800 Switch Series*. [Online]. Available: https://www.hpe.com/h20195/v2/GetPDF.aspx/4AA3-7115ENW.pdf
[41] BoNeSi. *The DDoS Botnet Simulator*. Accessed: 2018. [Online]. Available: https://github.com/markus-go/bonesi
[42] CAIDA Datasets. *Anonymized Internet Traces 2015*. Accessed: 2018. [Online]. Available: https://data.caida.org/datasets/passive-2015
[43] CAIDA Datasets. *DDoS Attack 2007*. Accessed: 2018. [Online]. Available: https://data.caida.org/datasets/security/ddos-20070804
[44] [Online]. Available: https://www.wireshark.org

TRUNG V. PHAN received the B.S. degree in electronics and telecommunications from the Hanoi University of Science and Technology, Vietnam, in 2015, and the M.S. degree in information communication technology from Soongsil University, Seoul, South Korea, in 2017. His research interests include software-defined networks, network functions virtualizations, and network security.

MINHO PARK received the B.S. and M.S. degrees in electronics engineering from Korea University, in 2000 and 2002, respectively, and the Ph.D. degree from the School of Electrical Engineering and Computer Science, Seoul National University, Seoul, South Korea, in 2010. He was with Samsung Electronics, from 2002 to 2004. As a Postdoctoral researcher, he has been with Carnegie Mellon University (CMU), since 2011, for two years. He is currently an Assistant Professor with the School of Electronic Engineering, Soongsil University, Seoul. Before a Postdoctoral Researcher with CMU, he was a Senior Engineer with the 3GPP LTE S/W Development Group, Samsung Electronics. His current research interests include wireless networks, vehicular communication networks, network security, and cloud computing.