A Comprehensive and Effective Mechanism for DDoS Detection in SDN

Mauro Conti, Ankit Gangwal
Department of Mathematics
University of Padua, Italy
Email: {conti, ankit.gangwal}@math.unipd.it

Manoj Singh Gaur
Department of Computer Science & Engineering
Malaviya National Institute of Technology, India
Email: gaurms@mnit.ac.in

***Abstract*—DDoS attack is one of the major concerns for network and cloud service providers, due to its substantial impact on revenue/cost and especially on their reputation. Also, network administrators are looking for solutions to manage voluminous data traffic. SDN is an emerging networking paradigm that provides a flexible network management. Hence, SDN is being widely adopted for wired, wireless, and mobile networks. Apart from a single point of failure (the controller), an attacker can target SDN at various levels by DDoS attacks.**

Existing solutions either focus on a particular attack type or require cumbersome alterations in SDN infrastructure. In this paper, we propose a comprehensive, yet effective and lightweight approach to detect various fundamentally different DDoS attacks in SDN. Our approach relies on sequential analysis. We employ a non-parametric change point detection technique called Cumulative Sum (CuSum). Our framework also includes an adaptive threshold scheme that adapts with the changing traffic pattern. Additionally, our framework can be tuned to suffice critical security requirements such as high detection rate and low false alarm rate. We evaluated the effectiveness of our solution using CAIDA Internet traces as well as DARPA intrusion detection evaluation dataset. Our results confirm the effectiveness of our mechanism. In particular, average false alarm rate in our experiments was under 11.64%. On average, our method is able to detect DDoS attacks within 4.15 seconds.

***Index Terms*—Distributed Denial of Service (DDoS), Software Defined Networks (SDN), Network Security**

I. Introduction

In today's Internet, end hosts have almost no control over the quantity or type of traffic forwarded to them. Typically, Internet Service Providers (ISPs) are responsible for regulating the traffic in network through traffic engineering. An ISP must take into account several important factors when performing the traffic engineering tasks, including highly unpredictable and dynamic nature of the Internet traffic, its resources and their capacity, its Service Level Agreements (SLA) with its customers, its policies and agreements with other ISPs. Moreover, an ISP would never want to upset its consumers by dropping their traffic despite the fact that a substantial amount of the traffic may be potentially unwanted by a consumer.

Such quandary of an ISP leads to several significant problems, especially to the catastrophic Distributed Denial of Service (DDoS) attacks that not only affect end hosts but sometimes also affect the ISP itself. Since their inception, DDoS attacks are still one of the biggest threats to the Internet's stability and security. With the increase in capacity of the Internet, the scale of DDoS attacks has also enlarged. As an illustrative example, a hosting company OVH was the victim of a 1 Tbps DDoS attack that hit its servers, which was one of the largest attacks ever seen on the Internet till late 2016[1]. Such attacks have been partially facilitated by user-friendly tools, e.g., Low Orbit Ion Cannon (LOIC) [1], hping3 [2], Stacheldraht [3]. Such tools enable even novice users to launch massive attacks against several targets simultaneously. Furthermore, employment of techniques such as IP spoofing makes it even harder to track and identify the attacker.

Recent developments and innovations in networking assure to change how the future Internet will work. In particular, networking infrastructure along with data plane and control plane witnessed promising improvements. The data plane is typically responsible for packet forwarding while the control plane takes all the routing decisions. Figure 1 depicts a simplified architecture of the conventional network, where the control plane and the data plane are embedded into the same device. In general, the forwarding rules are hardwired into a traditional device, and hence, traditional networks lack flexibility. Traditional networks are largely un-programmable by their owners while the innovations are limited to vendors or their partners. Besides, the devices have longer hardware fabrication cycles, and network management remains complex [4].

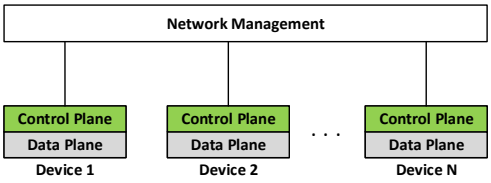

Figure 1: A simplified architecture of the traditional network

Software Defined Network (SDN) is a recently proposed networking architecture that completely separates the control plane from the data plane. All the networking elements in the data plane act as a simple packet forwarding device while all the routing decisions are made by a logically centralized system, i.e., the controller in the control plane [5]. Figure 2 presents a simplified architecture of SDN. The programmability of the control plane enables us to devise resilient routing logics, which can accommodate diverse requirements of various network applications.

[1] http://securityaffairs.co/wordpress/51640/cyber-crime/tbps-ddos-attack.html

978-1-5386-3839-2/17/$31.00 ©2017 IEEE

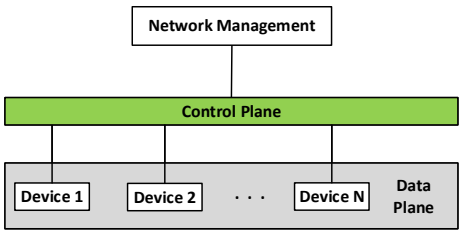

Figure 2: A simplified architecture of SDN

The potential of SDN has gained enormous attention from the research community as well as the industries. As SDN is still an emerging networking concept, it has various concerns related to performance, reliability, management, and security [6]. The performance issues include utilization of switch resources (e.g., bandwidth, flow-table size), efficient handling of new flows, and lookup procedures. The management issues mainly focus on careful management of the control plane and its resources as the control plane is responsible for handling the entire network. The reliability issues comprise link failure, controller failure, and asynchronous update of switches. However, the main objective of our work is to analyze the security issues in SDN; DDoS attacks in particular, and propose an effective DDoS detection framework.

Contributions: In this paper, we introduce a comprehensive, effective, and lightweight approach for detection of DDoS attacks in SDN. The major contributions of our work are listed as follows:

1) We thoroughly analyze how DDoS attacks impact on the different levels of SDN architecture.
2) We propose a simple, yet efficient solution for the detection of DDoS attacks in SDN.
3) We emulated our solution and evaluated its effectiveness using Internet traces provided by CAIDA [7] as well as DARPA intrusion detection evaluation dataset [8].

Organization: The remainder of this paper is organized as follows. Section II thoroughly explains various important aspects of DDoS attacks. Here, we discuss the typical classes of DDoS attacks, DDoS detection techniques, and how DDoS attack can be launched in SDN environment along with its repercussions on SDN. Here, we also present an overview of related works regarding DDoS attacks in SDN. In Section III, we elaborate the threat model. In Section IV, we explain our detection approach with its implementation details. Section V covers the details of our experimental setup while we discuss and analyze our results in Section VI. Finally, Section VII concludes the paper and explores the future directions.

II. Preliminaries and Related Work

DDoS is a cyber-attack where the attacker uses more than one machine (usually compromised) to make network services or resources unavailable to its intended users. A DDoS attack is typically launched in four phases namely: recruit; exploit; infect; and use [9]. In the recruit phase, the attacker scans remote machines for security holes that will help to barge in. In the exploit phase, the discovered loopholes are exploited to break into vulnerable machines. Such machines are then infected with the attack code in the next phase. And finally, the compromised machines are used to launch attack payload. Based on the targeted protocol level, DDoS attacks can be broadly classified into two categories [10, 11]:

1) Transport/network-level attacks: Such attacks use ICMP, TCP, UDP, and DNS protocol packets to launch DDoS. The aim here is to disrupt legitimate users' connectivity by exhausting the bandwidth of victim's network. The attacker can either use direct flooding or reflection-based flooding. In reflection-based flooding, the attacker sends forged requests to a large number of hosts, which in turn reflects massive replies towards the target.
2) Application-level attacks: Such attacks focus on exhausting server's resources such as CPU, memory to interrupt legitimate users' services. In general, the attacker employs request flooding attacks and slow request/response attacks.

The remainder of this section explains the typical DDoS detection techniques, DDoS attack scenario in SDN environment along with its impact on SDN architecture, and state-of-the-art regarding DDoS detection in SDN.

A. DDoS Detection Techniques

DDoS detection techniques can be widely classified into two categories: signature-based detection, and anomaly-based detection.

Signature-based Detection: A signature is a pattern of string that corresponds to a known threat or attack. The signature-based detection relies on string comparison techniques. Such methods compare and search for the current unit of activity such as a packet entry or a log entry in a signature repository. Signature-based detection approaches are efficient to identify only recognized attacks without any complex procedures. On another side, such methods are not capable of identifying variants of known attacks as well as new attacks. Other challenges include keeping an up-to-date signatures repository and proliferating size of the signature database.

Anomaly-based Detection: As opposed to signature-based detection, anomaly-based detection does not require predefined signatures or patterns to classify an activity. Such methods employ statistical features of network traffic to identify attacks. As a representative example, incoming packet rate can serve as a feature. The current network behavior is compared with the observed network behavior, and an alarm is raised when there is a significant variation from the normal course of operation. Such methods are capable of identifying variants of known attacks as well as unknown attacks. Nevertheless, they may create various spurious alarms [12].

B. DDoS in SDN

A DDoS attack in SDN environment can affect the network at various levels. It is worth mentioning that some attack vectors are common to the traditional network while some

threats are unique to SDN. For instance, instead of high-volume traffic flows, an attacker might use low-volume traffic flows to generate a huge number of *Packet_In* messages, which in turn overloads the ingress switch as well as the controller. Figure 3 shows that the attacker can strike on the switch, the controller, and the secure link between the control plane and the data plane.

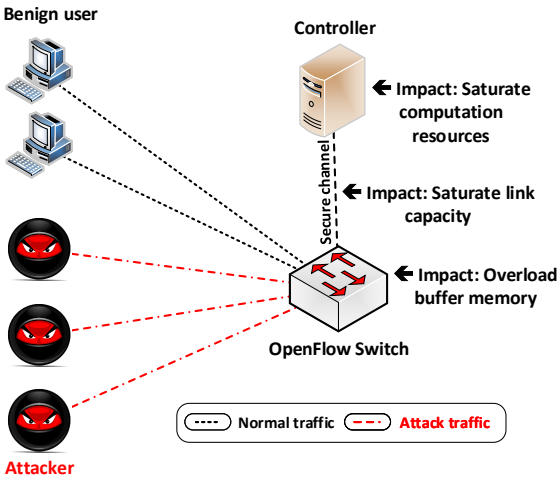

Figure 3: Impact of DDoS attack in SDN

Impact on switches: The objective of the attacker is to drop or at least delay legitimate users' packets to deteriorate the performance of the network and ultimately, ruin users' experience. Normally, when a new flow of packets reaches a switch, a "table-miss" event is raised. Consequently, a *Packet_In* message is forwarded to the controller to obtain an action. For all subsequent packets in the same flow, the switch applies the obtained action without any controller intervention. While the switch awaits for a response from the controller, it buffers the incoming packets in its buffer memory. In case the buffer gets full, the subsequent packets are dropped due to insufficient space in the buffer. In general, the attacker employs IP spoofing to generate a huge number of flows with random headers. Hence, botnets overflow the switching device with supposedly fresh flows of packets. Consequently, packets from legitimate hosts are dropped or at least delayed.

Impact on controller: The logically centralized controller is the single point of failure, and its breakdown can disrupt the entire network. The controller computes an action set for each *Packet_In* request coming from a switch. Calculating the action sets consumes controller's resources such as CPU, memory, I/O bandwidth. The controller can handle a large, but still a limited number of request at a given instance of time. Hence, with a huge number of requests generated by a DDoS attack will saturate the resources of the controller. Eventually, genuine requests are delayed or even dropped.

An attacker may also seek to interfere with controller's functioning through attempts such as buffer overflow, which may lead to the installation of erroneous forwarding rules in the data plane.

Impact on secure channel between control and data plane: The control and the data plane communicate over a secure channel. The secure channel carries periodic as well as sporadic messages. Even minute congestions in the channel may lead to inevitable delays in network functioning. Especially, delaying *Packet_In* messages notably degrades network performance. An enormous number of flows generated by a DDoS attack can saturate the secure channel, eventually ceasing the operation of the entire network [13].

C. DDoS Detection in SDN

Several researchers argued that a sensible solution for DDoS attacks is to enhance the security of every Internet host and prevent the damages from such attacks [14], while others suspect the widespread acceptance of such mechanisms [15]. Other researchers insisted that DDoS attacks are not even a security issue, but they are scalability questions [16]. In support of their claim they say that the attackers will continuously attempt to make their requests indistinct from the benign traffic; hence, they may defeat the detection mechanisms. In this scenario, the final solution is to increase resources in terms of quantity, which is expensive.

Mousavi et al. [17] proposed an entropy-based mechanism to detect a DDoS attack in SDN. Here, in case of an attack, the entropy decreases on the basis of the randomness of incoming packets' destination address. However, the approach assumes that the number of hosts in the network will always remain static and destination IP addresses are always evenly distributed for normal traffic. Mehdi et al. in [18] used maximum entropy estimation to determine benign traffic distribution and anomaly detection in SDN. The solution focuses only on small networks such as office and home networks.

Braga et al. [19] performed cluster analysis to detect DDoS in SDN. In this work, the system continuously collects statistical features of the flows and observe the collected features to identify any unusual activity. However, gathering and observing a large amount of data significantly deteriorates the performance of the controller. YuHunag et al. in [20] proposed a flow monitoring system to identify a DDoS attack. The proposed approach produces spurious alarms in a situation when a benign user starts to generate a large volume of traffic. Dong et al. in [21] introduced a solution to deal with *Packet_In* flooding attack against the controller. Shin et al. in [22] proposed a system called Avant-Guard that identifies DDoS attack induced by a flooding of TCP SYN packets. LineSwitch [23] improves Avant-Guard through a solution based on probability and blacklisting. However, both Avant-Guard and LineSwitch focus only on SYN flooding-based saturation attacks.

Kotani et al. in [24] proposed a *Packet_In* filtering approach for protection of control plane in OpenFlow networks. The solution requires large TCAM space to accommodate pending flow tables. Also, it fails when the datapath cannot parse packets of certain protocols and extract the required information, for instance, payloads in ARP, VLAN ID in 802.1Q headers. To defend against DDoS attacks, SDNShield [25] requires

deployment of specialized software boxes. Wang et al. [26] introduced a DDoS mitigation architecture where the DDoS mitigation strategy relies on a public cloud provider, which takes actions against the threats. However, the authors did not clarify where the suspicious traffic is hosted after it is detected.

Kalliola et al. [27] proposed a machine-learning based approach that integrates traffic learning with external blacklist information for DDoS detection. The defense mechanism works at the IP layer; hence, attacks targeting other layers do not fall within the scope of the proposed defense mechanism. The work presented in [28] employs a multi-controller system to solve the problem of DDoS attacks. However, the approach has several limitations. On one side, it employs random packet transmission delay to protect from scanning attacks, which in fact, affects the data transmission for legitimate users. On another side, synchronization of prolonged route tables among multiple controllers is overlooked.

To summarize, existing solutions either focus on a particular attack type or require alterations in SDN infrastructure and support from external entities such as public cloud provider. Our work is different from the state-of-the-art on various dimensions: (1) it can detect various fundamentally different attacks; (2) it does not require any change to the infrastructure or support from external entities; (3) it does not need any exhaustive training before implementation; and (4) it adapts automatically to changing traffic pattern.

III. Threat Model

The target of the attacker is a server, i.e., the victim server, which provides services to the hosts. The victim's network employs SDN as an underlying network architecture. The attacker has no information about the topology of victim's network, but the attacker knows that the victim's network is using SDN. The attacker and the victim may reside in the same or different networks. The attacker could be an individual user, a group of users, or a group of compromised systems (bots) controlled by the attacker. The attacker uses IP spoofing as a camouflage technique. When the attacker launches a DDoS attack (for attack details, please refer to Section V), it reaches victim's network and can cause damages to the network resources as well as to the victim.

IV. Proposed Approach

In this section, we present our framework for detection of DDoS attacks in SDN. Here, we explain the fundamental principles, followed by its comprehensive implementation details.

A. Cumulative Sum (CuSum)

A non-parametric method, called the CuSum approach, is an anomaly detection technique used for change point detection. CuSum based mechanisms measure the deviation of current observation from a historical (long-term) average of the observations. In our framework, we consider the volume of packets flowing per unit of time as a parameter to CuSum. When the current observation overshoots the historical average of the observations, then the value of CuSum coefficient ascends, and vice versa. Hence, if the value of CuSum coefficient surpasses an implemented threshold, it designates an exaggerating packet arrival rate, which is likely to be due to a DDoS attack [29]. Equation (1) shows the computation of the CuSum coefficient:

$$S(t) = \max\{0, (S(t-1) + N_{pk}(t) - m(t))\}; \quad S(0) = 0, \quad (1)$$

where t represents the time of current observation, $t-1$ represents the time of previous observation, $S(t)$ represents the CuSum coefficient at time t, $N_{pk}(t)$ represents the number of packets arrived between $t-1$ and t, and $m(t)$ represents the long-term average of packets arrived till t. Equation (2) shows the computation of $m(t)$:

$$m(t) = \epsilon * m(t-1) + (1-\epsilon) * N_{pk}(t); \quad m(0) = 0, \quad (2)$$

where the value of ϵ varies from zero to one, i.e., $0 < \epsilon < 1$. It is clear from Eq. (2) that a value of ϵ that is greater than 0.5 indicates dominance of the historical average of packet count; otherwise, current packet count holds more importance.

Interpreting a CuSum graph is straightforward. A segment of the CuSum graph with a positive slope represents a duration when the values tend to be higher than the overall average. Similarly, a segment of the CuSum graph with a negative slope represents a duration when the values tend to be lower than the overall average. An abrupt change in the direction of the CuSum value represents a sudden change or shift in the average. Segments where the CuSum graph observes a relatively straight path represents a period when the overall average did not change much.

Need for an Adaptive Threshold: Our approach incorporates an adaptive threshold system for the following reasons:

1) A static threshold cannot consider the tendencies and recurring conduct of the network traffic. As an example, traffic load during peak hours is expected to remain higher as compared to off-peak hours, which may induce abundant false alarms if a static threshold is engaged.
2) On the contrary, an adaptive threshold can adapt to the trends of traffic.
3) An adaptive threshold can help to reduce false alarms [30].

B. CuSum with Adaptive Threshold

Our method uses an adaptive threshold for CuSum with the following rules:

For the current value of CuSum (C_L) in a window (W_L) of L seconds:

1) **if** $C_L > \mu_{C_L} + k \cdot \sigma_{C_L}$ **then**
   $threshold_{new} = threshold_{old} + \alpha \cdot threshold_{old}$
2) **else if** $C_L < \mu_{C_L} - k \cdot \sigma_{C_L}$ **then**
   $threshold_{new} = \max(threshold_{old} - \beta \cdot threshold_{old}, min\_threshold)$
3) **else**
   $threshold_{new} = threshold_{old}$

where, k is a constant, α and β determines the degree of adjustment in the threshold. The values of α and β can be chosen to create a slow-increase/fast-decrease effect.

Here, μ_{C_L} represents the average of CuSum values in W_L, which is computed using Eq. (3).

$$\mu_{C_L} = \frac{\sum_{i=1}^{L} C_i}{L}. \quad (3)$$

And σ_{C_L} represents the standard deviation of CuSum values in W_L, which is computed using Eq. (4).

$$\sigma_{C_L} = \sqrt{\frac{\sum_{i=1}^{L}(C_i - \mu_{C_L})^2}{(L-1)}}. \quad (4)$$

C. System Model

We implemented our framework as an SDN controller module that works alongside other forwarding modules. The value of CuSum for the server is computed periodically. At the same time, μ_{C_L} and σ_{C_L} of the CuSum values within the window W_L are also computed. The window of observation glides in a sliding window fashion to accommodate new values. The threshold is adjusted according to the rules described in Section IV-B. When the observed CuSum overshoots the threshold, then an attack is identified.

One important aspect in CuSum computation is to enumerate the packets arrived between $t-1$ and t, i.e., $N_{pk}(t)$. The OpenFlow [31] switches maintain counters that include the number of packets, bytes, etc., for each flow entry. In our approach, the controller utilizes the built-in message exchange capabilities of the OpenFlow protocol and proactively interacts with the switches. As a part of the proactive interaction, the controller periodically exchanges common *FLOW_STATS* messages to acquire real-time traffic statistics. Using *FLOW_STATS* replies from the switches, the controller can enumerate $N_{pk}(t)$. Each flow-rule has a *HARD_TIMEOUT* and a relatively shorter *IDLE_TIMEOUT*. A shorter *IDLE_TIMEOUT* ensures rapid eviction of momentary flow-rules.

V. Experiments

We built our test scenario as reliable and realistic as possible, by considering the suggestions in [32]. Figure 4 shows the network topology of our test scenario. The target system resides in Network 1, which is composed of three OpenFlow switches controlled by a POX[2] controller. Here, Open vSwitch[3] serves as an OpenFlow-enabled switch. The botnets that flood DDoS traffic are in Network 2 while legitimate traffic flows from hosts residing in Network 2 and Network 3. Inter-network links are configured with 1 Gbps bandwidth and 25 ms of delay while the links among OpenFlow switches are configured with 1 Gbps bandwidth. All other intra-network links are set to 100 Mbps bandwidth. We emulated our test scenario using the Mininet[4].

[2]POX - `http://github.com/noxrepo/pox/`
[3]Open vSwitch - `http://openvswitch.org/`
[4]Mininet - `http://mininet.org/`

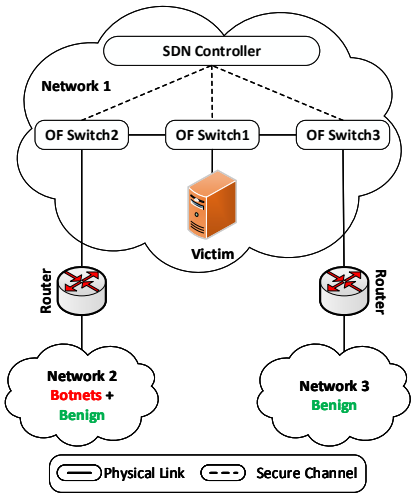

Figure 4: Test scenario's topology

As SDN is still an emerging networking concept, no DDoS attack dataset for SDN was publicly available at the time of evaluation. To evaluate our proposed approach, we orchestrated two different experiments. In the first experiment, we produced synthetic traffic for both legitimate users and botnets using Scapy[5]. To differentiate between the attack traffic and the normal traffic we used traces provide by CAIDA. These network traces are longer than an hour in duration and contain a distribution of traffic for various geographical locations. The information available in CAIDA traces was used to compute the average packet transmission rate for a benign source in a network. After considering the work presented in [33], we constituted a 20% attack traffic where botnet traffic comprised ICMP pings and had a rate higher than the normal traffic.

To assess effectiveness and versatility of our proposed mechanism we set another experiment. Here, we used DARPA intrusion detection evaluation dataset as they contain ample type of attacks, i.e., over 200 instances of more than 50 types of attacks. We downsampled and transformed the packet traces to match our emulation settings. As a representative example, Table I shows some attacks that may overload various components of an SDN environment.

Attacks	Descriptions
Smurf	Victim's source IP is used to broadcast ICMP requests to a network, which creates a reply flood towards the victim.
Neptune	A flood of SYN on one or more TCP ports.
IPsweep	ICMP pings are sent to every address within a subnet, and ping responses help to identify which hosts are listening.
Portscan	A surveillance sweep that scans several ports to identify which services are running on a machine.

Table I: Some attacks that may overload SDN components

It is important to note that these attacks have different working principles and they work at different layers. For example, "IPsweep" works at the network layer while "Neptune" works at the transport layer. Although "Portscan" and "IPsweep" are not considered as DDoS attacks by conventional intrusion

[5]Scapy - `http://www.secdev.org/projects/scapy/`

detection mechanisms, however, they may be used to generate a huge number of traffic flows to overload SDN components.

VI. Results and Analysis

In this section, we present and discuss the results from our experiments. Figures 5, 6, and 7 are plotted with respect to the first experiment mentioned in Section V. The purpose of Figure 5 and Figure 6 is to illustrate the effect of CuSum values on the threshold. Figure 5 depicts the computed value of CuSum and the corresponding threshold under the normal traffic. The initial threshold was computed according to the information available in the CAIDA traces. Although the traffic generator script generates the traffic right from the beginning, it takes a while to transmit the actual traffic flows. Because in the beginning, the hosts exchange initial network message such as ARPs. Once the network stabilizes, the normal traffic exhibits a relatively steady state. Meanwhile, the threshold adapts according to the value of CuSum. Since the threshold remained higher than the CuSum value, the traffic was classified as benign traffic.

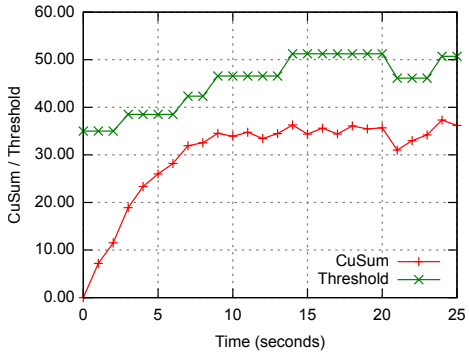

Figure 5: Variation of threshold against CuSum values under normal traffic

Figure 6 depicts the computed value of CuSum and the corresponding threshold under the attack traffic. After the initial exchange of network messages, attack flows generate a huge amount of traffic. The threshold adapts according to the traffic, but the value of CuSum quickly surpasses the threshold. The CuSum value exceeded the threshold around the sixth second and continued to stay above it.

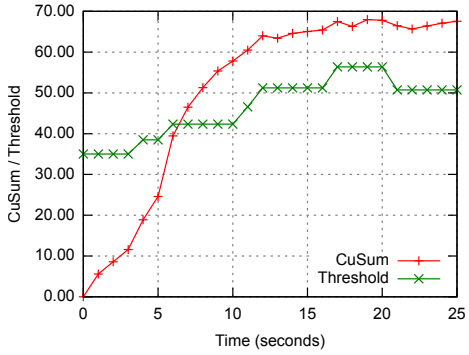

Figure 6: Variation of threshold against CuSum values under attack traffic

Figure 7 depicts the computed value of CuSum and corresponding threshold under the traffic that contains both normal and attack traffic. In this case, two attack sessions were scheduled, the first one starting from the nineteenth second till the twenty-ninth second and other one starting from the fifty-ninth second till the sixty-eighth second. Both the attack sessions were detected within four seconds. Although, there were a few false positives after the attack sessions were over.

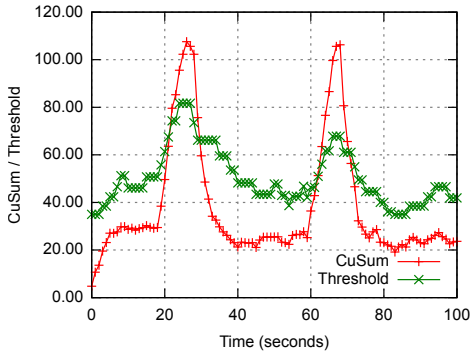

Figure 7: Variation of threshold against CuSum values under the traffic consisting of both normal and attack traffic

Now we discuss the results of the second experiment. Here, we conducted experiments for each combination of k, α, β, and W_L. The chosen values of k were 0.5, 1.0, 2.0 while the values for α, β were 0.100, 0.050, 0.025. For the window size (W_L), the chosen values were from 1 to 10. We used Detection Rate (DR), False Alarm Rate (FAR), Accuracy (ACC) to assess our proposed approach. The confusion matrix [34] as presented in Table II is a standard matrix that is widely used for the assessment of classification methods.

Confusion Matrix		**Predicted Label**	
		Normal	**Attack**
Actual Label	**Normal**	True Negative (TN)	False Positive (FP)
	Attack	False Negative (FN)	True Positive (TP)

Table II: Confusion matrix

DR measures the percentage of correctly identified attacks over all the actual attacks and is computed using Eq. (5).

$$DR\ (\%) = \frac{TP}{TP + FN} * 100. \tag{5}$$

FAR measures the percentage of legitimate traffic incorrectly identified as attack over the entire legitimate traffic and is computed using Eq. (6).

$$FAR\ (\%) = \frac{FP}{FP + TN} * 100. \tag{6}$$

ACC measures the percentage of true detection over the entire traffic trace and is computed using Eq. (7).

$$ACC\ (\%) = \frac{TP + TN}{TP + TN + FP + FN} * 100. \tag{7}$$

To illustrate the effectiveness of our approach, Figure 8, as an illustrative example, shows DR, FAR, and ACC for different window sizes with k set to 1 and α, β set to 0.025. In this case, since our method did not produce any false negative, DR was 100% for all window sizes. While FAR was 11.63% for window size one and two seconds, which improves and reaches 6.98% for wider windows. Since ACC is affected by true as well as false detections; hence, ACC was 90.38% for window size one and two seconds, which improves and reaches 94.23% for wider windows.

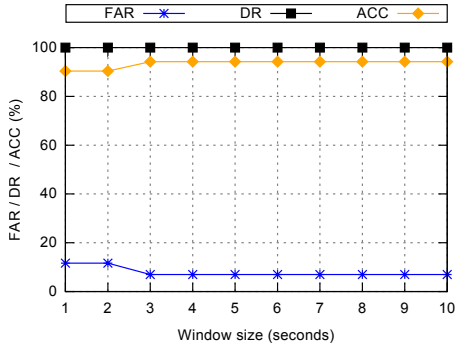

Figure 8: DR, FAR, and ACC for different window sizes where k=1 and α = β = 0.025

We performed ten experiments for each combination of k, α, and β, one for each window size. Due to space limitation, we show the results of the experiments where α was set equal to β. Table III presents the average and the standard deviation of DR, FAR, and ACC computed for results from each window size under every unique combination of k, α, and β. As explained in Section IV-B, the value of α, β decides the degree of modification in the threshold.

k	$\alpha = \beta$	μ_{DR} (%)	σ_{DR} (%)	μ_{FAR} (%)	σ_{FAR} (%)	μ_{ACC} (%)	σ_{ACC} (%)
0.5	0.100	10.00	31.62	1.16	3.68	83.46	2.43
	0.050	74.44	20.98	1.86	3.60	94.04	3.07
	0.025	96.67	5.37	5.58	2.50	94.81	1.82
1.0	0.100	35.56	46.50	2.56	4.83	86.73	5.47
	0.050	94.44	5.86	4.88	3.71	95.00	2.43
	0.025	100.00	0.00	7.91	1.96	93.46	1.62
2.0	0.100	100.00	0.00	9.07	2.99	92.50	2.47
	0.050	100.00	0.00	11.16	0.98	90.77	0.81
	0.025	100.00	0.00	11.63	0.00	90.38	0.00

Table III: Average value and standard deviation of DR, FAR, and ACC for different values of k, α, β

DR degrades with increasing value of α, β because a larger value of α, β modifies the threshold more as compare to smaller values. Consequently, attacks are misclassified. While FAR increases with decreasing value of α, β because the threshold does not appropriately adapt for a smaller value of α, β. An increasing value of k improves the DR, while the FAR is also increased. ACC observes no direct relation with α, β, or k as it relies on both true and false detections.

Detection Time: Another important evaluation criteria for any detection method is the time it takes to detect the attack. The detection time in our approach improves with increasing window size. The detection time was under six and a half seconds for all the experiments, and the average detection time considering all the experiments was 4.15 seconds with a standard deviation of 1.92 seconds.

Overhead: In our solution, the controller utilizes commonly exchanged *FLOW_STATS* messages to obtain real-time traffic statistics (in particular, to compute $N_{pk}(t)$) for DDoS detection. The induced overhead of our solution majorly depends on the frequency of message exchange. To assess the overhead, we configured the controller to exchange messages every second on a system with Intel Core i5-7200U CPU @ 2.50 GHz x 4 processor. The measured CPU overhead due to message exchange was nearly 11%. We believe that the reported overhead is not an issue since in a real-network scenario, the controller runs on a dedicated resource-rich server grade system.

VII. Conclusion and Future Work

SDN provides a simpler network administration with more flexibility as compared to the traditional networks. There are several security-related concerns in SDN, which are still required to be solved. In this work, we have proposed a framework that is capable of detecting various fundamentally different DDoS attacks in SDN. As shown by the results, the proposed approach is not only effective, but it can also be tuned on various parameters to fit vast security requirements, e.g., high DR, low FAR.

In the future, we will extend our approach to find a feasible way to mitigate an attack after its detection. We would also extend our approach to multi-domain networks containing more than one SDN controller. It would be interesting to investigate the detection and mitigation of DDoS attacks where multiple SDN controllers can communicate with each other over dedicated (e.g., EAST/WEST bound) interfaces.

Acknowledgment

Mauro Conti is supported by a Marie Curie Fellowship funded by the European Commission (agreement PCIG11-GA-2012-321980). Ankit Gangwal is pursuing his Ph.D. with a fellowship for international students funded by Fondazione Cassa di Risparmio di Padova e Rovigo (CARIPARO). This work is partially supported by the EU TagItSmart! Project (agreement H2020-ICT30-2015-688061). This work is partially supported by the grant n. 2017-166478 (3696) from Cisco University Research Program Fund and Silicon Valley Community Foundation, and by the grant "Scalable IoT Management and Key Security Aspects in 5G Systems" from Intel. This work is also partially funded by the project CNR-MOST/Taiwan 2016-17 "Verifiable Data Structure Streaming".

References

[1] A. M. Batishchev, "Low Orbit Ion Cannon (LOIC)," https://sourceforge.net/projects/loic, 2012.
[2] S. Sanfilippo, "hping3," http://www.hping.org/, 2006.
[3] D. Dittrich, "The stacheldraht DDoS tool," http://staff.washington.edu/dittrich/misc/stacheldraht.analysis.txt.

[4] F. Hu, Q. Hao, and K. Bao, "A survey on software defined network and OpenFlow: From concept to implementation," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2181–2206, 2014.

[5] A. Akhunzada, E. Ahmed, A. Gani, M. Khan, M. Imran, and S. Guizani, "Securing software defined networks: Taxonomy, requirements, and open issues," *IEEE Communications Magazine*, vol. 53, no. 4, pp. 36–44, 2015.

[6] Y. Jarraya, T. Madi, and M. Debbabi, "A survey and a layered taxonomy of software defined networking," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 1955–1980, 2014.

[7] The CAIDA UCSD Statistical information for the CAIDA Anonymized Internet Traces, http://www.caida.org/data/passive/passive_trace_statistics.xml, 2016.

[8] MIT Lincoln Laboratory, "Intrusion detection attacks database," http://www.ll.mit.edu/ideval/docs/attackDB.html.

[9] M. Jelena and R. Peter, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004.

[10] 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.

[11] S. Ranjan, R. Swaminathan, M. Uysal, and E. W. Knightly, "DDoS-resilient scheduling to counter application layer attacks under imperfect detection," in *IEEE INFOCOM*, 2006, pp. 1–13.

[12] H. J. Liao, C. H. R. Lin, Y. C. Lin, and K. Y. Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[13] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes, "Software defined networking security: Pros and cons," *IEEE Communications Magazine*, vol. 53, no. 6, pp. 73–79, 2015.

[14] F. Kargl, J. Maier, and M. Weber, "Protecting web servers from distributed denial of service attacks," in *WWW*, 2001, pp. 514–524.

[15] M. Sachdeva, G. Singh, and K. Kumar, "Deployment of distributed defense against DDoS attacks in ISP domain," *International Journal of Computer Applications*, vol. 15, no. 2, pp. 25–31, 2011.

[16] Y. Chung, "Distributed denial of service is a scalability problem," *ACM SIGCOMM Computer Communication Review*, vol. 42, no. 1, pp. 69–71, 2012.

[17] S. M. Mousavi and M. St-Hilaire, "Early detection of DDoS attacks against SDN controllers," in *IEEE ICNC*, 2015, pp. 77–81.

[18] S. A. Mehdi, J. Khalid, and S. A. Khayam, "Revisiting traffic anomaly detection using software defined networking," in *RAID*. Springer, 2011, pp. 161–180.

[19] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS flooding attack detection using NOX/OpenFlow," in *IEEE LCN*, 2010, pp. 408–415.

[20] C. YuHunag, T. MinChi, C. YaoTing, C. YuChieh, and C. YanRen, "A novel design for future on-demand service and security," in *IEEE ICCT*, 2010, pp. 385–388.

[21] P. Dong, X. Du, H. Zhang, and T. Xu, "A detection method for a novel DDoS attack against SDN controllers by vast new low-traffic flows," in *IEEE ICC*, 2016, pp. 1–6.

[22] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "AvantGuard: Scalable and vigilant switch flow management in software defined networks," in *ACM CCS*, 2013, pp. 413–424.

[23] M. Ambrosin, M. Conti, F. De Gaspari, and R. Poovendran, "LineSwitch: Efficiently managing switch flow in software defined networking while effectively tackling DoS attacks," in *ACM ASIACCS*, 2015, pp. 639–644.

[24] D. Kotani and Y. Okabe, "A Packet-In message filtering mechanism for protection of control plane in OpenFlow networks," in *ACM/IEEE ANCS*, 2014, pp. 29–40.

[25] K. Chen, A. R. Junuthula, I. K. Siddhrau, Y. Xu, and H. J. Chao, "SDNShield: Towards more comprehensive defense against DDoS attacks on SDN control plane," in *IEEE CNS*, 2016, pp. 28–36.

[26] B. Wang, Y. Zheng, W. Lou, and Y. T. Hou, "DDoS attack protection in the era of cloud computing and software defined networking," *Elsevier Computer Networks*, vol. 81, pp. 308–319, 2015.

[27] A. Kalliola, K. Lee, H. Lee, and T. Aura, "Flooding DDoS mitigation and traffic management with software defined networking," in *IEEE CloudNet*, 2015, pp. 248–254.

[28] D. Ma, Z. Xu, and D. Lin, "Defending blind DDoS attack on SDN based on moving target defense," in *SecureComm*. Springer, 2014, pp. 463–480.

[29] I. Ozcelik, Y. Fu, and R. R. Brooks, "DoS detection is easier now," in *2nd GENI Research and Educational Experiment Workshop*, 2013, pp. 50–55.

[30] V. A. Siris and F. Papagalou, "Application of anomaly detection algorithms for detecting SYN flooding attacks," *Elsevier Computer communications*, vol. 29, no. 9, pp. 1433–1442, 2006.

[31] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[32] J. Mirkovic, S. Fahmy, P. Reiher, and R. K. Thomas, "How to test DoS defenses," in *Cybersecurity Applications & Technology Conference for Homeland Security*, 2009, pp. 103–117.

[33] J. Sommers, V. Yegneswaran, and P. Barford, "Toward comprehensive traffic generation for online IDS evaluation," *Technical Report, University of Wisconsin*, 2005.

[34] P. Dokas, L. Ertoz, V. Kumar, A. Lazarevic, J. Srivastava, and P. N. Tan, "Data mining for network intrusion detection," in *NSF Workshop on Next Generation Data Mining*, 2002, pp. 21–30.