TPDD: A Two-Phase DDoS Detection System in Software-Defined Networking

Yi Shen, Chunming Wu, Dezhang Kong, Mingliang Yang
College of Computer Science and Technology
Zhejiang University, Hangzhou, China
Email:{shenyizju, wuchunming, kdz,ml_yang}@zju.edu.cn

***Abstract*—Distributed Denial of Service (DDoS) attack is one of the most severe threats to the current network security. As a new network architecture, Software-Defined Networking (SDN) draws notable attention from both industry and academia. The characteristics of SDN such as centralized management and flow-based traffic monitoring make it an ideal platform to defend against DDoS attacks. When designing a network intrusion detection system (NIDS) in SDN, how to obtain fine-grained flow information with minimal overhead to the SDN architecture is a problem to be solved. In this paper, we propose TPDD, a two-phase DDoS detection system to detect DDoS attacks in SDN. In the first phase, we utilize the characteristics of SDN to collect coarse-grained flow information from the core switches and locate the potential victim. Then we monitor the edge switches located close to the potential victim to obtain finer-grained traffic information in the second phase. The collection method of each phase fully considers the impact on the bandwidth between the controller and switches. Without modifying the existing flow rules, the collection module can obtain sufficient information about traffic. By using entropy-based and machine learning-based methods, the detection module can effectively detect anomalies and identify whether the potential victim marked in the first phase is the target of attacks. Experimental results show that TPDD can effectively detect DDoS attacks with little overhead.**

I. Introduction

Distributed Denial of Service (DDoS) attack [1] is one of the major threats to the current network. Attackers utilize botnet to send a huge number of useless packets to victims, attempting to exhaust the limited resource of targets. According to [2], the longest attack in the second quarter of 2019 lasted 509 hours and was directly against Chinese telecom operator China Unicom, which also set a new record since the series of reports began. Besides, the duration of DDoS sessions continued to grow. The overall growth was due to the duration of technically complex attacks. Thus, how to detect and defend against DDoS attacks is an urgent problem to be solved.

With the rapid development of infrastructure virtualization and cloud computing, Software-Defined Networking (SDN) [3] has been promoted as a new network architecture and has been deployed in environments like cloud, data center, etc. By decoupling control and data planes, SDN can achieve a centralized and software-based network management plane [4]. As a result, SDN can acquire the global view of the network and achieve flow-based traffic monitoring and control. These characteristics of SDN make it an ideal platform to defend against DDoS attacks. However, SDN itself can be a target of DDoS attacks [5]. Considering the limited space in flow tables and bottleneck of the communication channel between the controller and data planes, we need to address the following challenges when monitoring the whole network:

1) How to obtain fine-grained flow information when utilizing the limited rule space in flow tables?
2) How to achieve traffic monitoring in the entire network while imposing minimal overhead to the communication between controller and switches?

As a network intrusion detection system (NIDS), collecting traffic information is the first step to detect anomalies. When designing a lightweight NIDS in SDN, researchers have to consider which switches should be monitored. If the controller needs to monitor every switch in the network, the overhead caused by traffic monitoring is high, especially in large-scale networks. Many researchers try to lighten the overhead by using additional devices and adding intelligence to OpenFlow switches. However, these methods are not compatible with the original idea of the OpenFlow protocol. Besides, it is tough to implement complicated classification algorithms in switches.

In this paper, we propose a two-phase detection system named TPDD to detect DDoS attacks in the SDN environment. Depending on the global view of the controller, we monitor the core switches to gather coarse-grained traffic information in the first phase (P1). The information collection methods in P1 neither need to add extra overhead to the communication channel between the controller and switches nor require modification to the existing rules in the flow table. We aim to locate the potential victim in P1. Then in the second phase (P2), we monitor edge switches located close to the potential victim to obtain fine-grained flow information. By adopting the flow-based traffic information collection method, entropy-based feature extraction, and classifiers based on SOM + k-NN and SVM, we can accurately identify whether the potential victim is the target of DDoS attacks.

The main contribution of our work can be summarized as follows:

1) We propose a two-phase detection scheme to detect DDoS attacks. By taking advantage of the characteristics of SDN and using sFlow-based flow information collection methods, the occupation of the bandwidth between controller and switches can be reduced, thus reducing the overhead

when achieving network monitoring.

2) Without modifying existing flow entries, our system can achieve traffic monitoring in the whole network and accurately detect DDoS attacks. After the detection, We discuss some available mitigation schemes to protect both the victim machines and the SDN controller.
3) We implement the detection system TPDD in the controller and validate the effectiveness using a publicly available dataset. The experimental results show our system can accurately detect attacks within seconds.

The rest of the paper is organized as follows. Section II discusses the related work. Section III introduces the design of our system architecture. Section IV presents the detailed system implementation. The experimental results are presented in Section V. Finally, we conclude the paper in Section VI.

II. Related work

In recent years, many detection mechanisms against DDoS attacks using SDN have been proposed by researchers. As an early study using SDN to detect DDoS, Braga et al. [6] made use of the flow statistics in OF switches to collect traffic information. By periodically sending requests to all OpenFlow switches, the controller can collect traffic information and extract flow features to detect traffic anomalies by Self-Organizing Map (SOM). However, this work did not consider the control plane overloading caused by massive request messages. To achieve fine-grained detection, Xu et al. [7] changed the flow monitoring granularities on all switches to locate victims and attackers. This method did not concern the overhead to the communication between the controller and switches either. Moreover, this method requires huge modification of flow rules in switches.

To cope with these problems, Giotis et al. [8] proposed to collect traffic information by combining OpenFlow (OF) and sFlow. sFlow [9] is a packet sampling technology for collecting traffic information. In addition, sFlow does not need to modify the flow entries in switches. However, the flow sampling may affect the accuracy of detection because not all packets are collected. FADM [11] used the OF-based collection method in small-scale networks, and used the sFlow-based method in large-scale networks. But, how to judge the network scale was not mentioned. Further, the deployment of sFlow agent requires careful consideration since it is unwise to deploy sFlow agent on every OF switch which inevitably adds overhead to the sFlow server.

Entropy was widely used to help detect DDoS attacks [8], [12]. However, using entropy alone cannot identify anomalies that do not disturb randomness. Utilizing machine learning-based methods in SDN is quite suitable because of the global view and programmability of SDN. In [6], [7], [10], Self-Organizing Map (SOM) is used to classify network states. However, as an unsupervised learning algorithm, SOM incurs high time consumption when performing the classification. Support Vector Machine (SVM) is used in [11] to classify network states because of its high accuracy. In [13], a modified decision tree algorithm is applied to detect DDoS attacks.

Many researchers proposed to reduce the overhead of the control plane by adding intelligence to the data plane [14], [15]. AVANT-GUARD [16] introduced a connection migration module and actuating triggers to help complete TCP connections, thus defending against threats like saturation attacks. Wang et al. [12] implemented an entropy-based DDoS detection algorithm in edge switches to reduce the heavy communication between the controller and switches. However, these approaches mentioned above need to use additional devices and extend the data plane, which is not compatible with the original idea of OpenFlow protocol. In our system, we propose to collect traffic information and make detection in the SDN controller, without extending the data plane.

III. System Design

A. Motivation of two-phase detection

For the convenience of description, we present a data center network with the tree topology in Fig. 1. Clusters based on fat-tree topologies scale well, thus using tree topologies is quite suitable in data centers [17]. In Fig. 1, core switches aggregate all traffic into the network, while edge switches locate close to hosts. There are two methods to monitor the network state: 1) monitoring core switches and 2) monitoring edge switches. Efficient flow-based DDoS attack detection requires the fine-grained flow monitoring scheme. Because of the limited space in the flow table, flow rules in core switches are generally aggregated, which makes it hard to directly gather the finest-grained flow statistics when monitoring core switches. By contrast, edge switches have less flows to process, so that flow rules deployed in edge switches can reach finer granularity. However, without any prior knowledge of which target will be attacked, the detection system needs to monitor every edge switch in the network. Therefore, the overhead of monitoring in this method is very high, especially when the network scale is large. To this end, we propose to use a two-phase detection scheme to detect DDoS attacks.

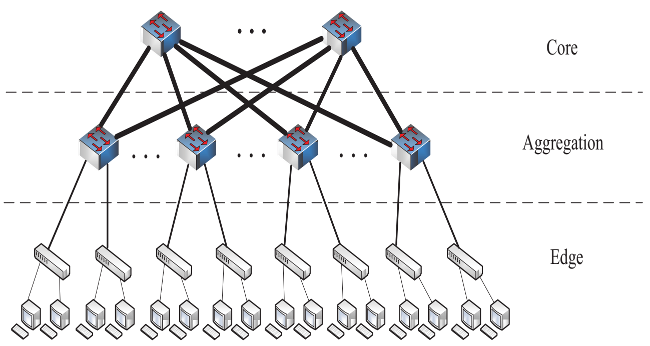


Fig. 1. Example of a tree topology

B. Design principles

The architecture of TPDD is shown in Fig. 2. In the first phase (P1), we monitor the core switches to detect anomalies

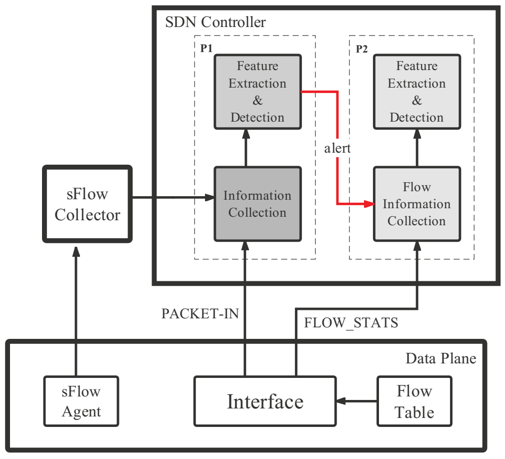


Fig. 2. System design

and find potential victims. When anomalies are detected, an alert will be triggered. After that, we monitor the edge switches located closest to victims to obtain fine-grained flow statistics to make accurate detection in the second phase (P2). The details of each component will be described in the next section.

Since the information collection of the detection module in P1 lasts throughout the life cycle of the SDN controller, we aim to lighten the overhead of collection methods when designing the detection module in P1. After the potential victim hosts are located, attacks can be more easily detected when observing traffic at the edge switches close to the potential victims. Thus in P2, we monitor the edge switches to achieve accurate detection. In addition, the detection system should be designed to ensure the real-time capability. In other words, the whole detection system should be lightweight under normal states and capable of achieving efficient and real-time detection when the DDoS attacks occur.

IV. System Implementation

A. Detection in P1

1) Information collection: In P1, the controller monitors the core switches to collect information about the whole network. According to the OpenFlow protocol, the table-miss flow entry in the flow table specifies how to process packets unmatched by other entries. Generally, the accordingly actions include sending to the controller by a Packet-In message, so that the controller can decide how to install rules for these new flows. The Packet-In message contains the packet header or the full packet. Thus, our collection module can extract the complete header information about unmatched packets from Packet-In messages without extra communicational overload.

sFlow is a sampling technology for network traffic monitoring. The sFlow mechanism consists of distributed sFlow agents deployed in OpenvSwitches and a centralized sFlow collector. sFlow agents collect the flow samples and counter samples and then send them to the sFlow collector. Thus the sFlow collector can gather flow information and traffic statistics from a centralized point. This sFlow-based monitoring method is utilized by many previous works [8], [11]. Because the flow samples cannot capture all packets, which inevitably impacts the accuracy, we only use the counter samples to gather the traffic information.

The collection method in P1, which consists of Packet-In based and sFlow-based approaches, imposes no additional consumption to the bandwidth between the controller and switches. Besides, both approaches do not need to modify original rules in the flow table.

2) Feature extraction: After the collection, the feature extraction module extracts network features that are beneficial for detection and then gathers them for the classification module. When the DDoS attack is launched, massive packets are sent to exhaust the resource of targets, resulting in the increase of traffic volume to the victim area. Thus we extract the statistical information pkt^t and $byte^t$ in the t time period from the counter samples of sFlow collector. In addition, massive packets sent by attackers may cause an increase of Packet-In messages because there are no matching flow entries for these new packets. Further, most of the new-coming flows are sent from botnet and to the victims, which results in the fact that the source addresses of packets are dispersed while the destination addresses of packets are concentrated. Thus, we use the entropy to measure the randomness of properties. High entropy value indicates a dispersed distribution, while a low value denotes a concentration of distribution. Every Packet-In message includes the head of the packet, which consists of $\{IP\_src, port\_src, IP\_dst, port\_dst, protocol\}$. Assume the total number of flows collected by Packet-In in the t time is X_t, $N_{IP\_src(i)}(t)$ represents the number of packets with the source IP address $IP\_src(i)$ in the t time period, the frequency $P_{IP\_src(i)}(t) = \{p_i^t | i = 1, 2, 3, ..., N\}$ can be calculated based on Equation 1. N represents the total number of source addresses in the t interval.

$$p_i^t = \frac{N_{IP\_src(i)}(t)}{X_t} \tag{1}$$

Then according to the definition of entropy calculation, the entropy of source IP address can be calculated by Equation 2. Similarly, the entropy of destination IP address $H_{IP\_dst}^t$ can also be calculated.

$$H_{IP\_src}^t = -\sum_{i=1}^{N} p_i^t \log(p_i^t) \tag{2}$$

When servers in the network are under DDoS attacks, the number of Packet-In messages (i.e. $N_{pkt\_in}^t$) will grow. Besides, a big part of the Packet-In messages are caused by malicious flows, which results in the entropy of the destination IP address decreased, and the entropy value of the source IP address increased. Thus we represent the feature vector in P1 as $V1^t = \{N_{pkt\_in}^t, pkt^t, byte^t, H_{IP\_src}^t, H_{IP\_dst}^t\}$ to judge the current network state. After that, we use the Min-Max Normalization to normalize the number of Packet-In messages $N_{pkt\_in}^t$ and the traffic volume data (i.e. pkt^t and

$byte^t$), and use Equation 3 to normalize the entropy value. The normalized range is (0,1]. When the classification result indicates the network may be suffering from attacks, we aim to find potential victims. Hosts can be regarded as potential victims if there is an increasing number of new flows and traffic sent to them.

$$H^{t'}_{IP\_src} = \frac{H^t_{IP\_src}}{logN} \tag{3}$$

B. Detection in P2

1) Flow collection: Since we have identified the potential victims in P1, we collect information from the edge switches that locate closest to the potential victims in P2. By periodically sending OFPT_STATS_REQUEST messages to the edge switches, the SDN controller can obtain detailed statistical information about the flows to the potential victims. Since the victims are the minority in the network, only several edge switches are monitored by the controller in this phase. Thus, the communication overhead can be reduced. Furthermore, the polling interval of requests k can be set to a small value to reduce the detection delay.

2) Feature extraction: As for the feature extraction module, we employ a 6-tuple feature for the detection module in P2. Since we have located the address of potential victim with IP address IP_{victim}, we obtain flows sent to IP_{victim}: F_t = $\{F_i | F_i.IP\_dst = IP_{victim}, i = 1, 2, \cdots, M\}$, where M represents the number of flows. Then we get the total number of active source IP address $N^k_{src\_ip}$ and port $N^k_{src\_port}$ during the k time period. If the host is the target of DDoS attacks, the number of active source address will increase.

We get the statistical results: FS = $\{FS_i | FS_i = (IP\_src_i,\ Port\_src_i,\ IP_{victim},\ byte\_count_i,\ pkt\_count_i), i = 1, 2, \cdots, M\}$. The $byte\_sum^k$ and pkt_sum^k can be calculated as follows:

$$byte\_sum^k = \sum_{i=1}^{M} byte\_count_i \tag{4}$$

$$pkt\_sum^k = \sum_{i=1}^{M} pkt\_count_i \tag{5}$$

When attack happens, the number of flows to the victim will increase and the distribution of each flow rate will be more dispersed. So similar to the calculation in P1, we measure the randomness of byte and packet count distribution by entropy, which is calculated by Equation 6 and 7. The normalization methods are the same as those used in P1.

$$H^k_{byte} = -\sum_{i=1}^{N^k_{src\_ip}} \frac{byte\_count_i}{byte\_sum^k} \log \frac{byte\_count_i}{byte\_sum^k} \tag{6}$$

$$H^k_{pkt} = -\sum_{i=1}^{N^k_{src\_ip}} \frac{pkt\_count_i}{pkt\_sum^k} \log \frac{pkt\_count_i}{pkt\_sum^k} \tag{7}$$

The feature vector in P2 consists of 1) $N^k_{src\_ip}$, 2) $N^k_{src\_port}$, 3) $byte\_sum^k$, 4) pkt_sum^k, 5) H^k_{byte}, and 6) H^k_{pkt}.

C. Classification

The DDoS detection module classifies the current state as either normal (-1) or abnormal ($+1$) based on the feature vector extracted from different phases. We choose 1) SOM + k-NN and 2) SVM as the classifier.

1) SOM: SOM is an unsupervised neural network model and is generally applied to achieve data clustering. Compared with supervised learning methods, SOM requires long training time, which makes it unsuitable to perform real-time classification. Thus, we use SOM as the dimensionality reduction method which maps the high-dimensional feature samples to the 2-dimension space. After that, we use k-NN to classify the mapped samples. Since k-NN works well in low-dimensional feature space and SOM can perform dimension reduction, combining SOM and k-NN can reduce the communicational cost and achieve real-time classification.

2) SVM: SVM is a popular supervised-learning based classifier and widely used for classification. By finding an optimal separating hyperplane to maximize the distances between different classes, SVM separates the input samples in the feature space. In addition, SVM performs well when trained with small dataset, which makes it suitable for anomaly detection.

We use both methods and compare their performance in our detection module.

D. Discussion on the mitigation

After the DDoS attack is detected, countermeasures should be taken to protect the network. As TPDD can find the victim hosts and gather information about flows to the victim in P2, the malicious traffic can be migrated or dropped after the detection. At the beginning of the mitigation stage, misjudgment should be considered because of flash crowd traffic. Mechanisms like white-list mentioned in [8], [11] can be utilized to solve this problem. If the address is recorded in the white-list, the traffic will be identified as legitimate traffic while the others will be considered malicious. After that, malicious traffic can be dropped by sending blocking rules to switches. This response method can work effectively to protect the victim hosts if the malicious flows are identified correctly. However, such an active processing method may also block legitimate traffic whose address is not recorded in the white-list.

Another relatively conservative method is to do rate-limiting. OpenFlow [22] provides the meter table and enables to implement rate-limiting for each flow. Therefore, the controller can utilize the meter table to limit the rate of suspicious flows. This method can help protect the hosts and reduce the impact on benign traffic whose address is not in the white-list.

Also, the mitigation module should deal with the massive packet-in messages to protect the controller and avoid exhausting the bandwidth between the controller and switches during the DDoS attacks. SDN controller can send Flow-Mod message to modify the action of table-miss entry and instruct switches to temporarily stop sending Packet-In messages caused by malicious flows to the controller.

V. Evaluation

A. Experimental setups

In order to objectively evaluate the performance of our system, we use the dataset provided by ISCX2012 [18] and build the experimental environment accordingly, which is shown in Fig. 3. We use Mininet to simulate the SDN network and implement our system based on the POX controller [19]. sFlow-rt [20] is used for collecting counter samples.

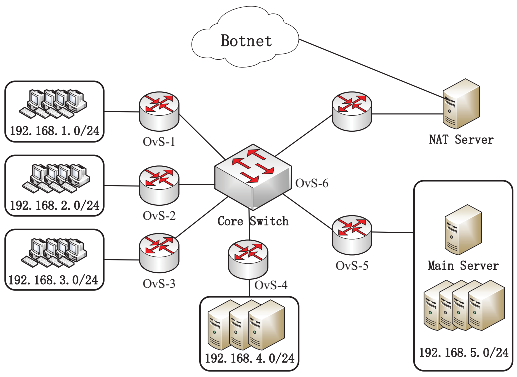


Fig. 3. Experimental topology

As Fig. 3 shows, 21 workstations are divided into 5 LANs. OvS-6 serves as the core switch that provides the final aggregation for the network. OvS-1, OvS-2, OvS-3, OvS-4, and OvS-5 serve as edge switches that located close to hosts. In OvS-6, flow rules are aggregated by IP_{dst} attribute. And in other edge switches, rules are installed with the finest granularity.

The UNB ISCX intrusion detection evaluation dataset consists of 7 days of network activity including benign traffic and malicious traffic. Besides, this dataset includes traffic with full packet payloads and labeled traces in pcap format. Tcpreplay [21] is used to replay the trace and adjust the rate. To emulate the DDoS attack, We extract the DDoS traffic from the dataset and replay it. Different from the attack scenario presented in [18], we replay the malicious traffic from the botnet rather than from the internal network. Meanwhile, we replay the benign traffic from hosts as the background traffic.

B. Detection model training

In order to train the detection model, we simulate the attack and normal scenario using the same rate as the one in the dataset. Then we extract network features from the flow collection module. After that, feature vectors are normalized and used to train the classifier. We collect the flow information every 5 seconds in P1 and every 3 seconds in P2. We present the number of training samples in Table I.

TABLE I
THE NUMBER OF TRAINING SAMPLES

	Attack samples	Normal samples
P1	258	341
P2	135	214

The map layer of SOM is a 20 $\times$ 10 matrix of neurons. The initial learning rate of SOM is 0.8 and the initial neighborhood radius is 5. Considering the number of training samples, we choose k = 7 in the k-NN algorithm.

C. Detection result

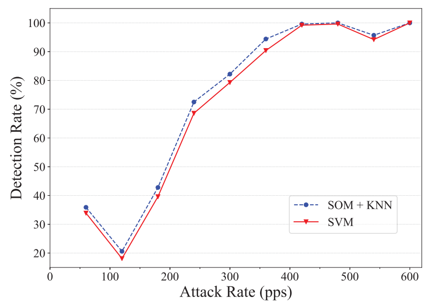
The effectiveness of our detection scheme is evaluated by the Detection Rate (DR) and the False Alarm rate (FA), which are computed using Equation 8 and 9, respectively.

$$DR = \frac{TP}{TP + FN} \tag{8}$$

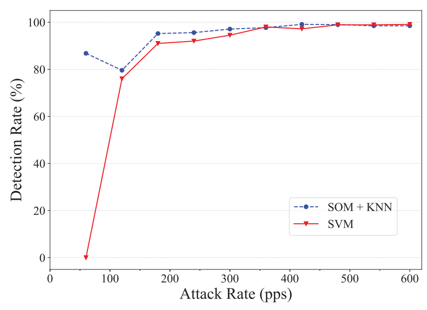
where TP (True Positives) are attack traffic that are correctly classified, and FN (False Negatives) are attack traffic that are detected as legitimate.

$$FA = \frac{FP}{TN + FP} \tag{9}$$

where FP (False Positives) are legitimate states that classified as malicious, and TN (True Negatives) are legitimate states that are correctly identified.



(a) Detection Rate in P1



(b) Detection Rate in P2

Fig. 4. Detection rate in different detection phases

As shown in Fig. 4, the detection rate of SOM + k-NN is slightly higher than SVM in both phases, and it performs better in P2 when the attack rate is less than 100pps. Comparing the accuracy of detection in different phases, the accuracy in P2 is higher than P1, which is because collecting fine-grained information about traffic is beneficial for detection.

In addition to that, we simulate another normal scenario using data extracted from another day in ISCX2012 to calculate the value of FA. FA has a value of 0 in P2 and no more than 5% in P1. When using SVM, the value of FA in P1 is 1.48%, and when using SOM + k-NN, the value is 2.22%. The experimental results indicate that the false alarm rates of both classifiers are very low.

We evaluate the real-time performance by measuring the detection delay between the start of attacks and the alerts. The average detection delay is presented in Table II. The results show that the detection delay of both classifiers is small.

TABLE II
AVERAGE DETECTION DELAY

	SVM	SOM + k-NN
P1	3.06s	3.05s
P2	6.5s	6.29s

D. System Overhead

We evaluate the overhead of TPDD by measuring the CPU utilization of the POX controller with and without TPDD in the normal scenario. As a result, the average CPU utilization of controller is 6.0% without TPDD and 7.6% with TPDD, which indicates that TPDD incurs little overhead under normal states. The main reason is that when no attack occurs, only the collection module of P1 works while the collection module of P2 is not started.

We also measure the CPU utilization of the POX controller in the normal environment, if every switch in the network is monitored by the controller. The polling interval in this setting is adjusted to 10 seconds so that the average detection delay can be set close to the delay of TPDD. Experimental results show that the average CPU utilization reaches 11.1%, which proves that the monitoring mechanism of TPDD is more lightweight than that of traditional methods when the detection delay of both systems are close.

VI. CONCLUSION

In this paper, we propose a two-phase detection system named TPDD to detect DDoS attacks in the SDN environment. In the first phase, we collect traffic information by collecting Packet-In messages and sFlow information. By monitoring core switches, we can identify the current network state and find the potential victim without modifying existing flow rules. Then in the second phase, we collect flow statistical information from edge switches located close to the victim to judge whether the potential victim is the target of attacks. We validate the effectiveness of TPDD with a public dataset. The experimental results show that TPDD can accurately detect DDoS attacks and protect the network with little overhead.

ACKNOWLEDGMENT

This work is supported by the National Key Research and Development Program of China (2018YFB1800601, 2018YFB2100404), the Key Research and Development Program of Zhejiang Province (2018C03052, 2019C01064, 2017C01064), and Major Scientific Project of Zhejiang Lab (2018FD0ZX01).

REFERENCES

[1] J. Mirkovic and P. Reiher, “A taxonomy of ddos attack and ddos defense mechanisms,” *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004.

[2] O. Kupreev, E. Badovskaya, and A. Gutnikov, “DDoS attacks in Q2 2019,” https://securelist.com/ddos-report-q2-2019/91934/

[3] 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.

[4] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, “A clean slate 4d approach to network control and management,” *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 5, pp. 41–54, 2005.

[5] 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 Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 602–622, 2015.

[6] R. Braga, E. de Souza Mota, and A. Passito, “Lightweight ddos flooding attack detection using nox/openflow.,” in *LCN*, vol. 10, pp. 408–415, 2010.

[7] Y. Xu and Y. Liu, “Ddos attack detection under sdn context,” in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pp. 1–9, IEEE, 2016.

[8] 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,” *Computer Networks*, vol. 62, pp. 122–136, 2014.

[9] P. Phaal, S. Panchen, and N. McKee, ”InMon Corporations sFlow: A Method for Monitoring Traffic in Switched and Routed Networks,” 2001.

[10] T. M. Nam, P. H. Phong, T. D. Khoa, T. T. Huong, P. N. Nam, N. H. Thanh, L. X. Thang, P. A. Tuan, V. D. Loi, *et al.*, “Self-organizing map-based approaches in ddos flooding detection using sdn,” in *2018 International Conference on Information Networking (ICOIN)*, pp. 249–254, IEEE, 2018.

[11] D. Hu, P. Hong, and Y. Chen, “Fadm: Ddos flooding attack detection and mitigation system in software-defined networking,” in *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pp. 1–7, IEEE, 2017.

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

[13] Y. Chen, J. Pei, and D. Li, “Detpro: A high-efficiency and low-latency system against ddos attacks in sdn based on decision tree,” in *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pp. 1–6, IEEE, 2019.

[14] S. K. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey, “Bohatei: Flexible and elastic ddos defense,” in *24th {USENIX} Security Symposium ({USENIX} Security 15)*, pp. 817–832, 2015.

[15] H. Wang, L. Xu, and G. Gu, “Floodguard: A dos attack prevention extension in software-defined networks,” in *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 239–250, IEEE, 2015.

[16] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, “Avant-guard: Scalable and vigilant switch flow management in software-defined networks,” in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 413–424, ACM, 2013.

[17] M. Al-Fares, A. Loukissas, and A. Vahdat, “A scalable, commodity data center network architecture,” in *ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 63–74, ACM, 2008.

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

[19] POX, Python-based OpenFlow Controller. http://www.noxrepo.org/pox/about-pox/

[20] InMon, sflow-rt. http://www.inmon.com/products/sFlow-RT. php.

[21] Tcpreplay, http://tcpreplay.appneta.com/

[22] Open Networking Foundation, “OpenFlow Switch Specification Version 1.3.0 (Wire Protocol 0x04), 2012.