Defending DDoS Attacks in Software-Defined Networking Based on Legitimate Source and Destination IP Address Database

Xiulei WANG, Ming CHEN, Changyou XING, Nonmembers, and Tingting ZHANG, Student Member

SUMMARY The availability is an important issue of software-defined networking (SDN). In this paper, the experiments based on a SDN testbed showed that the resource utilization of the data plane and control plane changed drastically when DDoS attacks happened. This is mainly because the DDoS attacks send a large number of fake flows to network in a short time. Based on the observation and analysis, a DDoS defense mechanism based on legitimate source and destination IP address database is proposed in this paper. Firstly, each flow is abstracted as a source-destination IP address pair and a legitimate source-destination IP address pair database (LSDIAD) is established by historical normal traffic trace. Then the proportion of new source-destination IP address pair in the traffic per unit time is cumulated by non-parametric cumulative sum (CUSUM) algorithm to detect the DDoS attacks quickly and accurately. Based on the alarm from the non-parametric CUSUM, the attack flows will be filtered and redirected to a middle box network for deep analysis via south-bound API of SDN. An on-line updating policy is adopted to keep the LSDIAD timely and accurate. This mechanism is mainly implemented in the controller and the simulation results show that this mechanism can achieve a good performance in protecting SDN from DDoS attacks.

key words: network security, DDoS attacks, software-defined networking, non-parametric CUSUM

1. Introduction

Software-defined networking (SDN) empowers network operators with more flexibility to program their networks. With SDN, network management moves from codifying functionality in terms of low-level device configurations to building software that facilitates network management and debugging. By decoupling the control plane from the data plane, SDN provides new approaches to solve many age-old problems such as traffic engineering [1], congestion control [2], load balancing [3] and network security [4], [5], [11], [20]. However, the security of SDN itself is still an open issue [6].

The DDoS attacks have been the main threats to the availability of Internet for decades. In DDoS attacks, the attacker sends a large number of fake connections to the victims by botnet which involves many dummy hosts. As a result, the resource of the victim will be exhausted and legitimate requests will be denied. Since DDoS attacks can greatly degrade the performance of the network and are difficult to detect, DDoS attack protection attracts more concern. Many previously proposed DDoS detection approaches rely on monitoring the volume of traffic that is received by the victim [28], [29]. The authors in [31] pointed out that these approaches do not provide a way to differentiate flash crowds from DDoS attacks and they considered the number of new source IP address appeared in data flow in unit time as observing feature. The source-end DDoS attacks is detected by the CUSUM algorithm according to the legal source IP database which is built by the historical trace [24]. However, high false-positive rate is led because the approach takes only the source IP address into account and it can only be deployed in the source-end network.

Although there are already many DDoS protection mechanisms in the traditional network, the defense of the DDoS attacks in SDN still needs to be paid more attention because the working principle of SDN is different from the traditional network [32]. The authors in [6] summarized security threats identified in SDN and pointed out that the DDoS attacks harm the availability of SDN both in control plane and data plane. In [7], a DDoS attack to fingerprint SDN networks and further launch efficient resource consumption was presented for the first time. These studies demonstrate that SDN brings new security issues that could not be neglected. The DDoS attacks in data plane will harm the control plane as well and the security of controller is a bottleneck of SDN. A feasible approach of source IP based filtering technique to defeat DDoS attacks was proposed in [8]. In the approach, a temp table which is used to store source IP address of forward packets from the switch is defined and each unique IP address has a counter to track the number of arrived packets. The DDoS addresses are detected by the characteristics of initiating less connections and generating less packets per connection in a constant duration. However the short flow can be easily mistaken as the attack flow in this mechanism and the detection delay is too long to protect the controller. In order to protect the control plane, Daisuke Kotani et al. [30] proposed a mechanism to filter out Packet-In messages without dropping important ones for network control. In the mechanism, switches record the values of packet header fields before sending Packet-In messages, which are specified by the controllers in advance, and filter out packets that have the same values as the recorded ones. This mechanism needs to extend OpenFlow’s standard mechanisms and add some functions to the SDN switches, which makes it hard to be deployed in...
a short time. The added functions will degrade the performance of switches and it is difficult for network operators or controller developers to decide which Packet-In message is important. The authors in [10], [11] took advantage of SDN to redirect abnormal flows to security middle boxes networks, which both protected the network and improved the utilization of middle boxes. All of the research shows that SDN brings new attack plane for DDoS attacks. It is necessary to research the characteristics of DDoS attacks in SDN and propose effective mechanisms to protect the data plane along with control plane.

Inspired by the research introduced in [7], we first measured and observed the DDoS attacks in a SDN testbed. The experiments results showed that the computation and storage resources of the controller and switches drastically changed when the DDoS attacks started. Based on the working principle of SDN and these observations, we draw the conclusion that the drastic degradation of the performance was caused by the burst of the fake flows launched by DDoS attacks. This conclusion is the main motivation of our defending mechanism which defends against DDoS attacks based on the legitimate source and destination IP address pairs database. The mechanism firstly establishes a legitimate source and destination IP address database by the normal historical traffic trace of the network. Secondly, the proportion of the new source-destination IP address pair is cumulated and analyzed based on the non-parametric cumulative sum (CUSUM) [12], [13] algorithm. The algorithm will raise an attack alarm if the cumulative sum value exceeds the detection threshold. According to the attack alarm signal, the DDoS attack flows will be filtered and redirected to a security middle box network [10], [11] for deeper analysis. Our approach is different from the approaches proposed in [8], [31] because we use the proportion of new source-destination IP address pair other than the number of new source IP address in the flow volume in unit time as the detection feature, which make the mechanism much more accurate. No matter deployed in the source-end or the destination-end network, this mechanism can efficiently protected the SDN network. Compared with the mechanism proposed in [30], our mechanism is lightweight, which does not need to modify the functions of switches. The performance of the switches will not be affected.

The rest of this paper is organized as follows. After analyzing and showing the effect of DDoS attacks in a SDN testbed in Sect. 2, Sect. 3 presents the main idea of the security mechanism. Section 4 describes the framework and implementation of the security mechanism. In Sect. 5, the security mechanism is evaluated through a series of experiments. Finally, Sect. 6 concludes the paper and discusses the future work.

2. Measurements and Observations

Essentially, since the control plane is separated from the data plane in a SDN network, the data plane will typically ask the control plane to obtain flow rules when the data plane sees new network packets that it dose not know how to handle. This reactive control mode enables network operators to control network efficiently. However it also brings serious problems. From the flow chart shown in Fig. 1 we can see that there is no security mechanism to check the legality of the flows and all unmatched packets in data plane will be sent to the controller. There are two potential threats [14].

- **Data Plane DoS.** The fake communication requests generated by dummy hosts will produce many useless flow entries which will take up much flow table space. When the flow tables are overflowed, the legal flow requests will be denied.
- **Control Plane DoS.** The source IP address spoofing technology is widely adopted in most sophisticated DDoS attacks. When there are DDoS attacks in data plane, there will be a burst of unmatched flow requests to the controller. The computation and storage resources of the controller will be occupied by these fake requests.

In current situation, it is very hard to evaluate the DDoS attacks to SDN in the Internet, as SDN is not widely deployed to many networks yet. Therefore, a testbed based on OpenFlow [9] technology is built in this paper.

From Fig. 2 we can see that the testbed includes 3 subnets: network1, network2 and network3. The network1 is a
SDN subnet including a target host with IP 10.0.1.8, 2 normal hosts and 3 OpenFlow switches controlled by the controller. The Tribe Flood Network 2000 [15] (TFN2K) is deployed in network2 to simulate DDoS attacks. Two high performance hosts located in network3 named generator1 and generator2 are deployed to generate background traffic. In order to simulate the background traffic, the dataset without any DDoS attacks supplied by Lincoln Laboratory [16] is analyzed. The frequency distribution histogram of flows per-second is shown in Fig. 3. By Kolmogorov-Smirnov test (KStest), we can see the number of flows per-second approximately follows Poisson distribution with parameter \( \lambda = 4.273 \). Secondly, the empirical distribution function of the number of flows and the number of packets per-seconds is shown in Fig. 4. From Fig. 4 we can see that 99% of the flow number per-second is less than 20 and 99% of the packet number per-second is less than 600.

Based on the analysis of the normal traffic dataset, the background traffic of the testing scenario is configured as follows.

1. The dataset is replayed at the rate of 600 packets per second by Tcpreplay [17].
2. Running 7 virtual machines in host1 with different IP address from 10.0.1.1 to 10.0.1.7. The traffic generator2 generates flows at the rate which obeys Poisson distribution with parameter \( \lambda = 1000 \). The destination IP address of each flow randomly set from 10.0.0.1 to 10.0.0.200 while the source IP address of each flow randomly distributes from 10.0.0.1 to 10.0.0.200. The length of each flow is set oscillating with a uniform distribution whose average value is 30 packets, minimum value is 20 packets and maximum is 40 packets.

The TFN2K is used to generate DDoS flooding attacks. There are two TFN2K servers and a TFN2K client located in network2 to simulate the botnet. The TFN2K will control servers to send attack traffic to target host and the IP address spoofing is used.

In order to evaluate the DDoS attacks in SDN, 3 metrics are defined as follows.

**Number of PACKET_IN events/s.** A PACKET_IN event will be sent to controller by underlying OpenFlow switches when a new flow arrives. The number of the PACKET_IN events has positive correlation with the number of new flows. The more PACKET_IN events the controller receives, the heavier the work load of the controller is.

**Number of flow entries in tables/s.** The controller will install flow entries into the flow tables of switches. The number of the flow entries has positive correlation with the number of new flows.

**Match/lookup ratio/s.** whenever a packet arrives at the OpenFlow switch, the OpenFlow switch will perform the lookup and match actions. Match/lookup ratio/s has the negative correlation with the number of new flows. All of the lookup and matched packets number will be recorded and updated.

The routing application which is implemented based on the shortest path algorithm is running on the controller named POX [18]. The switch is simulated by a host which runs the Open vSwitch [19]. All of the hosts in the testbed are equipped with Quad-Core 2.8GHZ Intel i5 CPU and 8GB RAM memory. The experiment last for 20 minutes. The UDP flood DDoS attack with spoofing source IP address will be launched by network2 at the 360th second and lasts for 10 minutes.

The number of PACKET_IN events received by controller per-second is shown in Fig. 5. Under normal traffic, the value oscillates around 200 and the maximum value is less than 500. When DDoS attack starts, the value rapidly increases to 2000 which is about 10 times larger than that under normal traffic.

The number of flow entries in OF switch2 is shown in Fig. 6. From the results we can see that the number is oscillating with 1000 and the maximum value is less than 1100. When DDoS attack starts, the value will dramatically increase from 1000 to about 7000 and the minimum value is...
larger than 6000.

The match/lookup ratio of OF switch$_2$ is shown in Fig. 7. From Fig. 7 we can see that when DDoS attack happens, the value will rapidly decrease from 85% to 20%.

We also measured the end-to-end communication delay and packet loss during DDoS attack. The results show that the delay is larger than 1000s and the packet loss is almost 100% when DDoS attack happens.

The number of the PACKET_IN events and the number of the flow entries have positive correlation ship with the number of new flows. The burst increasing of PACKET_IN events and flow entries number is caused by the burst increasing of new flows launched by DDoS attacks. As the flow table is overflow and most of the flows are new, the Match/lookup ratio/s will decreases sharply. The computation and storage resources of the control and data plane will be exhausted by the fake flows, so the end-to-end communication performance will be affected. This situation will be even worse under the highly distributed denial of service (HDDoS) with spoofing source IP address. It is necessary to design security mechanisms to protect SDN from DDoS attacks.

3. Motivation and Methodology

The DDoS attacks can be detected by the number of new flows emerged[8], [21]. A major drawback of this method is that a normal flash crowd may be mistaken as a DDoS attack. It has been found that the ratio of new source IP addresses in DDoS attacks is 86% to 99.4% while this value is 17.1% in flash crowd[22], [23]. The statistics of local area network traffic found that there is a high probability for the IP addresses which have been seen before to emerge again[22]. Let $\delta_{\text{norm}}$, $\delta_{\text{flash}}$ and $\delta_{\text{attack}}$ denote the number of flows in unit time received by the target network under normal, flash crowd and DDoS attacks respectively. Let $\theta_{\text{norm}}$, $\theta_{\text{flash}}$, $\theta_{\text{attack}}$ denote the proportion of new flows in unit time appeared under normal, flash crowd and DDoS attacks respectively. The relationship of these variables is shown in Eqs. (1) and (2).

\[ \delta_{\text{norm}} \ll \delta_{\text{flash}} \approx \delta_{\text{attack}} \] (1)

\[ \theta_{\text{norm}} \approx \theta_{\text{flash}} \ll \theta_{\text{attack}} \] (2)

The observations shown in Sect. 2 and Eqs. (1) (2) imply that the DDoS attacks can be detected according to the proportion of new flows appeared in unit time along with traffic volume. Each flow can be abstracted as a source-destination IP address pair and if the proportion of the source-destination IP address pair in unit time exceeds the threshold which is computed from the historical trace, the DDoS attack can be detected. In order to achieve this goal, there are two challenges need to be solved: constructing a legitimating source-destination IP addresses pair database and detecting the DDoS attacks as quickly as possible. The detail of building legitimating source-destination IP addresses pair database and the non-parametric CUSUM detection algorithm are described as follows.

3.1 Legitimate Source-Destination IP Addresses Database

**Definition 1:** LSDIAD (Legitimate Source-Destination IP addresses Database) denotes the collection of legitimate source-destination IP address pairs (LSDIP) of a network.

LSDIAD is built by off-line training and on-line updating. The off-line training adds LSDIP into LSDIAD according to the rules introduced in [24] via the historical trace.

As the network traffic gradually changes, there will be new LSDIP emerging. In order to keep the LSDIAD up to date, it needs to be updated on-line. We define a unidirectional source-destination IP address pairs as $sd = (sip, dip)$, where $sip$ and $dip$ denote the source and destination IP address of a unidirectional flow. Let $T$ denotes the update cycle of LSDIAD and $T$ is divided into $k$ equal time interval $\tau$. $T = \sum \in k \tau_i$, $\tau_1 = \tau_2 = \ldots = \tau_k$. $SD_i$ is a collection of $sd$ which appeared in $\tau_i$. At the end of every update cycle, the $sd$ which satisfies the following two rules $\chi_1$ and $\chi_2$ will be added to LSDIAD.

$\chi_1(d, T, \tau):$ The $sd$ which has appeared in at least $d$ different $SD_i$ during the update cycle $T$.

$\chi_2(u, T, \tau):$ The $sd$ which has exchanged at least $u$ packets during every time interval $\tau$ during the update cycle $T$.

If the LSDIAD is full, the least recently used $sd$ will be deleted. The $sd$ will also be collected during each time slot $\Delta$. We assume $\Delta_1 = \Delta_2 = \ldots = \Delta_\alpha$. Let $\phi_i$ represents the collection of unique $sd$ appeared during $\Delta_i$ and $\varphi_i$ denotes the collection of $sd$ both in $\phi_i$ and LSDIAD during $\Delta_i$. $|\phi_i - \phi_i \cap \varphi_i|$ denote the number of new $sd$ in $\Delta_i$. As $|\phi_i - \phi_i \cap \varphi_i|$ varies at different $\Delta$, the value can be normalized by defining $X_i = (|\phi_i - \phi_i \cap \varphi_i|)/\phi_i$. $X_i$ can be used as the random
sequence for detection.

3.2 Non-Parametric CUSUM Detection Algorithm

CUSUM algorithm works based on the fact that the probability of a random process will change if there is a change happened. Suppose one is able to sequentially collect a series of independent and identically distributed (iid) random observations, \(X_n\) for cases that regard the risk of making a false detection.

After the changepoint, subject to a tolerable limit on the risk of making a false detection, the challenge is to do so with as few observations as possible when the observations’ common distribution has changed. There is a shift between the observations, \(Z_t\), such that \(X_1, X_2, \ldots, X_t\) are each distributed according to normal distribution \(N(0, 1)\), while \(X_{t+1}, X_{t+2}, X_{t+3}, \ldots\), each adhere to normal distribution \(N(\delta, 1)\). We assume \(\beta = v\) and \(v\) (i.e., the changepoint) is assumed unknown non-random number; for cases that regard \(v\) as a random variable, see, e.g., [25], [26]. One’s aim is to detect that the observations’ common distribution has changed. The challenge is to do so with as few observations as possible following the changepoint, subject to a tolerable limit on the risk of making a false detection.

The problem is to sequentially differentiate between the hypotheses \(H_t: v = t, 0 < t < \infty\) (i.e., that the data \(X_n\) change their statistical profile at time instance \(v = t\), \(0 < t < \infty\)), and \(H_{\infty}: v = \infty\) (i.e., That no change ever occurred). To test \(H_t\) and \(H_{\infty}\), one first constructs the corresponding likelihood ratio, which for the iid scenario has the form

\[
L_{n,v} = \frac{\prod_{i=1}^{v} \phi(x_i) \prod_{i=v+1}^{n} \phi(x_i - \delta)}{\prod_{i=1}^{n} \phi(x_i)} = \frac{\prod_{i=v+1}^{n} \phi(x_i)}{\prod_{i=v+1}^{n} \phi(x_i)} \exp\left(\delta \sum_{i=v+1}^{n} \left(\frac{x_i - \delta}{2}\right)\right)
\]

and \(\prod_{i=v+1}^{n} \phi(x_i) = 1\), \(\sum_{i=v+1}^{n} x_i = 0\). Take the nature logarithm of \(L_{n,v}\), we have

\[
\Lambda_{n,v} = \ln L_{n,v} = \delta \sum_{i=v+1}^{n} \left(\frac{x_i - \delta}{2}\right)
\]

If there is a shift between \(x_1, x_2, \ldots, x_t\) and \(x_{t+1}, x_{t+2}, x_{t+3}, \ldots\), based on the maximum likelihood principle, the corresponding log likelihood ratio statistic is

\[
\Lambda_n = \max_{1 \leq v \leq n} \Lambda_{n,v} = \max\left(\delta \sum_{i=v+1}^{n} \left(\frac{x_i - \delta}{2}\right)\right)
\]

If we assume the detection is upward shift, \(\delta > 0\), the above-mentioned log likelihood ratio statistic equivalent to the following statistic:

\[
Z_n = \max_{1 \leq v \leq n} \sum_{i=v+1}^{n} \left(\frac{x_i - \delta}{2}\right)
\]

Suppose there is no mean shift before the \(n - 1\) observations, \(Z_i < h\) for \(i = 1, 2, \ldots, n - 1\), and \(h\) is threshold. The following criterion shown in Eq. (7) is used to decide if there is an upward shift at the \(n\)-th observation.

\[
\begin{align*}
Z_n &> x_n - \delta/2 > h \\
or Z_n + x_{n-1} - \delta > h \\
or Z_n + x_{n-1} + x_{n-2} - 3\delta/2 > h \\
&\vdots \\
or Z_n + x_{n-1} + \cdots + x_1 - n\delta/2 > h
\end{align*}
\]

Let \(\bar{x}_n = x_i - \delta/2\) and \(\bar{x}_0 = 0\), the CUSUM statistic in Eq. (6) can be described in Eq. (8) where \(\bar{S}_0 = 0\) and \(\bar{S}_k = \frac{1}{k} \sum_{i=0}^{k} \bar{x}_i\).

\[
Z_n = \max_{1 \leq v \leq n} \left\{ \left(\sum_{i=0}^{v} \bar{x}_i - \sum_{i=0}^{v} \bar{x}_i\right) = \bar{S}_n - \min \bar{S}_v \right\}
\]

As \(\bar{x}_n = \bar{S}_n - \bar{S}_{n-1}\) and \(\min \bar{S}_v = \min \{\bar{S}_n, \min \bar{S}_{n-1}\}\),

\[
Z_n - Z_{n-1} = \bar{x}_n - \min\left\{\bar{S}_n - \min \bar{S}_{n-1}\right\}
\]

\[
Z_n - Z_{n-1} = \bar{x}_n - \min\left\{\bar{S}_n - \min \bar{S}_{n-1}\right\} = \max\left\{\bar{x}_n, \bar{S}_n - \min \bar{S}_{n-1}\right\}
\]

Setting the uncertain parameter \(k = \delta/2\), the recursion formula of \(Z_n\) is:

\[
Z_n = \max\{0, Z_{n-1} + x_n - k\}, \ n = 1, 2, \ldots
\]

The CUSUM algorithm is described from Eq. (3) to Eq. (10) and it is decided by \((k, h)\) where \(k\) is called reference value and \(h\) is called decision boundary. If \(Z_n > h\) \((Z_i \leq h, i = 1, 2, \ldots, n - 1)\), there will be an alarm of attack.

Through the description of CUSUM we can see, it is necessary to obtain the parametric model of the random series which is useful to monitor the random sequence according to the probability density function. Unfortunately, Internet is a very dynamic and complicated entity, and the theoretical construction of Internet traffic models is a complex open problem, which is beyond the scope of this paper. Since non-parametric methods are not model-specific, they are more suitable for analyzing Internet. The non-parametric CUSUM algorithm achieves the real time detection by monitoring the input random variables in a sequential manner and accumulating the value of random observations which are significantly higher than the mean level under normal operation, which is more suitable for network.

Suppose \(x_n\) represents the proportion of the new source-destination IP address pair in the \(n\)-th time slot \(A_n\). The modified CUSUM algorithm will be shown from Eq. (11) to Eq. (16).

\[
\delta_n = (1 - \beta) \times \delta_{n-1} + \beta \times x_n, \ \delta_0 = x_0
\]

\[
Z_n = x_n - \delta_n - d
\]

\[
S_n = \sum_{i=0}^{n} Z_i, \ \ S_0 = 0
\]
\[ Y_n = S_n - \min_{1 \leq k \leq n} S_k \]  

(14)

Here \( \delta_n \) is the mean value of the statistical sequence \( \{x_i, \ i = 1, 2 \ldots n\} \), \( \beta \) denotes the exponentially weighted moving average (EWMA) coefficient, which typically ranges from 0.01 to 0.03. One of the assumptions for the nonparametric CUSUM algorithm is that mean value of the random sequence is negative during normal conditions, and becomes positive when a change occurs, thus \( d \) is a shift which makes the statistic \( E(Z_k) \) smaller than 0 under the normal condition and \( d \) is a constant value for a given network condition.

For a random sequence \( \{x_i\} \), if and only if \( Y_n \leq h \), \( n = 1, 2 \ldots i \) and \( Y_i > h \), it claims that there is no abnormal in the \( i-1 \) time slots and the abnormal is detected in the \( i \)-th time slot. This is called the modified CUSUM algorithm and for the efficiency of computing, the recursive form is described as follows.

\[ Y_n - Y_{n-1} = Z_n - \min\left\{0, S_n - \min_{1 \leq k \leq n-1} S_k\right\} \]
\[ = \max\left\{Z_n, Z_n - S_n + \min_{1 \leq k \leq n-1} S_k\right\} \]
\[ = \max\left\{Z_n, \min_{1 \leq k \leq n-1} S_k - S_{n-1}\right\} \]
\[ = \max\{Z_n, -Y_{n-1}\} \]
\[ Y_n = f(Y_{n-1} + Z_n), \quad Y_0 = 0 \]
\[ f(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases} \]

(15)

The decision function is shown in (16).

\[ d_N = \begin{cases} 0, & Y_n \leq N \\ 1, & Y_n > N \end{cases} \]

(16)

where \( N \) is the attack threshold and \( d_N \) represents the decision at time \( n \). If \( Y_n > N \), it denotes that there is an attack happened, otherwise no attack happened.

4. Framework and Implementation

In SDN, the controller could passively respond to various events from the underlaying data plane as well as proactively send commands to the specific equipments to get kinds of information such as flow statistics information, link state and queue length etc. The applications running on controller could manage and control network via the standard API between control plane and data plane, such as OpenFlow. Based on the architecture of SDN, a security mechanism is implemented as applications in the application plane of SDN.

Figure 8 presents the framework of the security mechanism. Most of the functions are implemented as 6 important modules in the application plane of SDN, including LSDIAD offline training module, LSDIAD online updating module, DDoS detection module, DDoS filter module, flow cache table updating module and PACKET_IN event handler module.

Before introducing these modules, the parameters used are defined in Table 1.

The offline training module adds LSDIPs into the LSDIAD offline based on normally historical traffic trace according to the rules introduced in Sect. 3. The network administrators can also add LSDIPs into LSDIAD according to the practical situation.

The flow cache table is used to store the information of unidirection flows. The information which mapped to flowi in flow cache table is defined as itemi = (sipi, dipi, pkt_numi, timesi), where pkt_numi denotes the number of packets exchanged from sipi to dipi. The timesi denotes the emerging frequency of the unidirection flow.

The cache table updating module will update the flow cache table by common Northbound API supplied by SDN. The update cycle of flow cache table is \( \tau \). At the end of every update cycle \( \tau \), the pkt_num of flowi will be updated by flow statistics API supplied by SDN.

At the end of every update cycle \( T \), LSDIAD will be updated by on-line updating module. If there is no attack alarm from detection module, the sd of the item in flow cache table which satisfies \( \chi_1 \) and \( \chi_2 \) will be added to LSDIAD. If the DDoS attack is detected, the on-line update will be suspended until the alarm = 0 again.

The process of PACKET_IN event handler module is shown in Table 2.

When an unmatched unidirection flowi arrives, the information of the flow will be extracted: flowi(sip, dip)
Algorithm 1: PACKET_IN handler

**Input:** PACKET_IN events, PACKET_IN_counter, alarm, matched_counter

1: \( \text{flow} \rightarrow \text{sip, dip} \) \(-\) extract (PACKET_IN_event)
2: \( \text{PACKET_IN_counter} ++ = 1 \)
3: if (flow[dip] in DDoS_target_IP):
   4:     forward to middle box network (flow, duration)
5: else:
   6:     result \(-\) lookup_LSDIAD(flow[si], flow[dip])
   7:     if result = True:
   8:         matched_counter = 1
   9:     install_flow_databoth(IP, PACKET_IN_event)
10: else:
   11:     result, item \(-\) lookup_flow_cache_table(flow[si], flow[dip])
   12:     if result = True:
   13:         item[times] = 1
   14:     install_flow_databoth(PACKET_IN_event)
15: else:
   16:     item = (flow[pkt_num] = 0, times = 1)
   17:     insert_flow_cache_table(item)
   18:     install_flow_databoth(PACKET_IN_event)

\( \rightarrow \) extract (PACKET_IN_event) in line 1. The PACKET_IN_counter will add 1 in line 2. If the destination IP \( \text{flow} \}_{[\text{dip}] is in the DDoS_target_IP, the flow will be directed to security middle box networks for deep analysis by forward to middle box network (flow, duration) in line 4. If \( \text{flow} \}_{[\text{dip}] is not in DDoS_target_IP, the flow will firstly be searched in the LSDIAD by lookup_LSDIAD(flow, [si], flow, [dip]) in line 6. If the source-destination IP address pair (\( \text{sip}, \text{dip} \)) is founded, the value of matched_counter will be increased by 1 and the path of flow will be installed from line 7 to line 9. If (\( \text{sip}, \text{dip} \)) is not founded, the flow will be secondly searched in the flow cache table by lookup_flow_cache_table(flow, [si], flow, [dip]) to check if the flow has emerged before. If flow is found in flow cache table, the frequency of the flow will be added by 1 and the data path will be installed from line 11 to 14. If the flow is new, we just insert the flow item into the flow cache table by insert_flow_cache_table(item) from 16 to 18. The DDoS_target_IP will be flushed when there is no attack.

At the end of each cycle \( \Delta \), two concurrent actions of the detection module will be executed.

1. The DDoS detection module sorts the items in flow cache table by pkt_num field and the flows with an abnormal number of packets will be detected. This step is effective for detecting naive DoS attack launched with large traffic volume. If the abnormal flow is detected, the parameter alarm will be set to 1.

2. The detection module executes the non-parametric CUSUM algorithm to detect if there is a DDoS attacks. The detection process contains 5 steps which are shown as follows.

   **Step 1:** Read matched_counter and PACKET_IN_counter from PACKET_IN event handler module and calculate \( x_n, x_0 = 1 - (\text{matched_counter}/\text{PACKET_IN_counter}) \).

   **Step 2:** Calculate \( \delta_n \) according to Eq. (11) and \( Z_n \) by Eq. (12).

   **Step 3:** Compute the cumulative sum \( Y_n \) by Eqs. (13)\(-\) (15).

   **Step 4:** Set matched_counter \( = 0 \), PACKET_IN_counter \( = 0 \).

   **Step 5:** If \( Y_n > \text{threshold} N \), set alarm = 2.

If any of the 2 kinds of DoS happens, the parameter will be send to the filter module. The filter function of DDoS attacks is shown in Table 3.

Unique destination IP addresses will be collected from flow cache table to dip_set from line 2 to line 4. For every dip in dip_set, the number of the items whose destination equals dip will be calculated from line 5 to line 8. The dip in dip_set which emerged the most times is recognized as the target IP of DDoS attacks and it will be added to the DDoS_target_IP from line 9 to line 11. All of the flows whose destination IP address equals the DDoS_target_IP will be deleted from the data plane by delete_flow() in line 14 and the flow cache table will be flushed in line 15.

### Table 2: The process of PACKET_IN event handler

| Line | Description |
|------|-------------|
| 1    | \( \text{flow} \rightarrow \text{sip, dip} \) \(-\) extract (PACKET_IN_event) |
| 2    | \( \text{PACKET_IN_counter} ++ = 1 \) |
| 3    | if (flow[dip] in DDoS_target_IP):
|      | 4:     forward to middle box network (flow, duration) |
| 5    | else:
|      | 6:     result \(-\) lookup_LSDIAD(flow[si], flow[dip]) |
|      | 7:     if result = True:
|      | 8:         matched_counter = 1 |
|      | 9:     install_flow_databoth(PACKET_IN_event) |
|      | 10: else:
|      | 11:     result, item \(-\) lookup_flow_cache_table(flow[si], flow[dip]) |
|      | 12:     if result = True:
|      | 13:         item[times] = 1 |
|      | 14:     install_flow_databoth(PACKET_IN_event) |
|      | 15: else:
|      | 16:     item = (flow[pkt_num] = 0, times = 1) |
|      | 17:     insert_flow_cache_table(item) |
|      | 18:     install_flow_databoth(PACKET_IN_event) |

### Table 3: The filter algorithm of DDoS

Algorithm 1: DDoS filter

**Input:** alarm, flow_cache_table, default_path

1: if alarm = 2:
   2:       for item in flow_cache_table:
   3:           if item[dip] \( \notin \) dip_set:
   4:               dip_set \( \leftarrow \) item[dip]
   5:       for dip in dip_set:
   6:           if item[dip] \( \notin \) DDoS_target_IP:
   7:               DDoS_target_IP \( \leftarrow \) dip:
   8:               dip_set \( \leftarrow \) dip:
   9:       if dip[num] = maximum and dip not in DDoS_target_IP:
   10:          DDoS_target_IP \( \leftarrow \) dip:
   11:          for item in flow_cache_table:
   12:              if item[dip] \( \in \) DDoS_target_IP:
   13:                 delete_flow(item[dip])
   14:     delete all items of the flow_cache_table
   15:     alarm = 0

5. Evaluation

In order to evaluate the feasibility and performance of the security mechanism proposed in this paper, we implemented all of these functions in the controller POX. The prototype of the security mechanism is deployed in the testbed built in Sect. 2.

LSDIAD is the base of the security mechanism. The operation efficiency of LSDIAD will greatly affects the establishing time of the flow data path. The security mechanism proposed in this paper is mainly deployed at the enterprise or local area SDN network. The scale of the local area network ranges from hundreds to thousands. The existing sophisticated database technology can meet the requirements. The flow cache table is implemented by existing mature hash table technology [27]. The communication trace without any DDoS is collected and the LSDIAD is initialized by the trace dataset according to the rules introduced in [24]. The building of the LSDIAD is similar with the method introduced in [24] and the performance of the method has already been evaluated. As the off-line building of LSDIAD has little effect on the protection of DDoS attacks, we mainly focus on evaluating the performance of DDoS detection and protection in this paper.
5.1 Performance of DDoS Detection Module

The DDoS attack is launched by sending a large amount of fake traffic to the target host in a short time. It is better to detect the attack as quickly as possible before the disastrous consequence. In order to evaluate the performance of the detection module, the experiment is designed as follows.

1. A new flow generator is added to the testing scenario. The generator can generate flows at the rate which obeys Poisson distribution with parameter $\lambda = w$. The destination IP address of each flow is 10.0.1.8 and the source IP address is randomly distributed from 10.1.0.1 to 10.1.0.200.

2. The experiment lasts for 10 minutes. The normal background traffic is generated from 0 to the 10th minute. The new flow generator starts to generate flows from the 5th minute.

3. The parameters of the modified CUSUM algorithm are set as follows: $\beta = 0.01$, $\Delta = 1s$, $d = 0.01336$ and $N = 0.4$. The value of $d$ and $N$ is trained according to the normal background traffic. We adjust the value of $d$ and $N$ until observing the desired results.

Figure 9 presents the detection performance of the detection module under different $w$. The statistics $x_n$ denotes the proportion of new flow emerged in the controller.

It is obvious that when the new flow generator starts to work, the value of $x_n$ will increase and the value of $Y_n$ will increase from 0 until it exceeds the threshold. When the value of the $w$ is larger, the value of $x_n$ and $Y_n$ increases drastically. When $w = 100$, the detecting time delay of the detection algorithm is about 3s which is much larger than 17s under the situation $w = 10$. As the average rate of new flow in DDoS is much larger than 100, the detection function could detect the attack less than 1s. From the Fig. 9 we can also see that larger threshold will increase false positive ratio of the detection function. In order to improve the accuracy of the detection function, the value of the threshold could be determined by the normal traffic of the edge network. In general, 100% of the DDoS attack will be detected when $N \in [0.4, 0.8]$.

5.2 Performance of Filter Module

In order to evaluate the performance of filter module, the experiments is designed as follows.

1. The experiment lasts for 20 minutes. The normal traffic generator1 and normal traffic generator2 simulate the normal traffic from 0 minute to 20th minute. The TFN2K client starts the DDoS attack to the target host from the 347th second to the 1000th second.

2. The parameters of the detection module are set as follows: $\beta = 0.01$, $\Delta = 1s$, $d = 0.01336$ and $N = 0.4$.

3. We repeat three different experiments with the parameter duration of the filter module separately set with 100s, 300s and 1200s respectively.

4. The number of PACKET_IN events, flow entries of switch tables and the match/lookup ratio per-second is collected and shown in Fig. 10, Fig. 11 and Fig. 12.

Figure 10 illustrates the number of the PACKET_IN events received by the controller per-second. The average number of the PACKET_IN events per-second will decrease greatly with the value of the duration increasing from 100s to 1200s. From Fig. 10 we can learn that when DDoS attack begins, the number of the PACKET_IN events will periodically change from about 200 to 2000. The cycle of the changes equals to the value of the parameter duration. This is because the value of duration equals to the lifetime of the
Table 4 The RTT and packet loss with different duration value

| duration | 30s  | 100s | 300s  | 1200s |
|----------|------|------|-------|-------|
| RTT      | 116.578ms | 64.154ms | 4.675ms | 4.032ms |
| Packet loss | 2%  | 1%   | 0%    | 0%    |

flow entries which filter the DDoS attack flows. When the lifetime of the flow entries expires, all of the attack flows will be forward to the controller again. The detection module will detect the abnormal and the filtering flow entries will be installed again. This process will repeat on and on until the DDoS attack terminates.

The number of the active flow entries in flow tables is shown in Fig. 11. The average number of the entries decreases with the increasing of the duration. The number of the flow entries per-second in flow tables is periodically changes from about 1000 to 4000. The cycle equals to the value of duration. Figure 12 shows the match/lookup ratio of the flow tables. The value of match/lookup ratio oscillates around 83% under the normal traffic. When DDoS begins, the value of match/lookup ratio periodically changes from about 100% to 20%. The cycle of the change equals to the value of the duration. The reason of results shown in Fig. 11 and Fig. 12 is that the filter flow entries of the DDoS attack flows will timeout. The lifetime of the filter flow entries equals to duration.

The other two import performance metrics to evaluate the performance of the security mechanism are round-trip time (RTT) and the packet loss of the end-to-end communication. Under normal background traffic without any DDoS attacks, the average value of RTT is about 3.947ms and the packets loss is 0%. Under the DDoS attack without any security mechanism, the average value of RTT is larger than 100s and the packet loss is 100%. The average value of RTT and packet loss of the security mechanism proposed in this paper under the DDoS attack traffic is shown in Table 4.

From Table 4 we can see that the average value of RTT and packet loss of the end-to-end communication will be greatly deduced under the DDoS attack. When the value of duration setting larger than the attacking time of DDoS, the RTT nearly equals to the value under normal traffic.

From the results of our experiments we can see that the security mechanism proposed in this paper can greatly avoid the harmfulness of HDDoS to the data plane and control plane.

6. Conclusion

The research described in [7] presented the potential DDoS attacks in SDN for the first time. Motivated by the research, we summarize the data plane DoS and control plane DoS by analyzing the working process of SDN and a testbed is built to evaluate the effect of DDoS attacks. Based on the observations, a security mechanism based on the legitimate source-destination IP address pairs is proposed to defend DDoS attacks in SDN. Based on LSDIAD, the non-parametric CUSUM algorithm can quickly and accurately detects DDoS based on the proportion of the new flows received by controller. The DDoS attack filter will be triggered when the DDoS attack is detected and then DDoS flows will be filtered and rerouted to a middle boxes network for deeper processing. Experimental results based on the testbed shows that the security mechanism proposed in this paper can greatly improve the performance of SDN under DDoS attacks. The LSDIAD is the basic of the mechanism, the construction of the LSDIAD will be the main work in the future. The more efficient and rational rules will be studied. The parameters of the non-parameter CUSUM algorithm are trained off-line which is not suitable for highly dynamic networks, so the self adapting method of the non-parameter CUSUM will be researched.

Acknowledgements

This work was supported by the State Key Development Program for Basic Research of China under Grant No. 2012CB315806, Nation Natural Science Foundation of China under Grant Nos. 61402521 and 61103225.

References

[1] S. Jain, A. Kumar, S. Mandal, et al., “B4: experience with a global-Deployed Software Define WAN,” Proc. SIGCOMM’13, Hong Kong, China, Aug. 2013.
[2] M. Ghobadi, S.H. Yeganeh, and Y. Ganjali, “Rethinking end-to-end congestion control in software-defined networks,” Proc. ACM Workshop on Hot Topic in Networks, 8 (4), pp.61–66, 2012.
[3] N. Handigol, S. Seetharaman, M. Fajislak, N. McKeown, and R. Johari, “Plug-n-Server: load-balancing web traffic using OpenFlow,” Proc. SIGCOMM 2009 (Demo), Barcelona, Aug. 2009.
[4] M. Casado, M.J. Freedman, J. Pettit, J. Luo, N. Gude, N. McKeown, and S. Shenker, “Rethinking enterprise network control,” IEEE/ACM Trans. Netw., vol.17, no.4, pp.1270–1283, 2009.
[5] S.A. Mehdi, J. Khalid, and S.A. Khayam, “Rethinking traffic anomaly detection using software defined networking,” Proc. 14th international conference on Recent Advancements in Intrusion Detection, vol.9, pp.161–180, Springer-Verlag, 2011.
[6] D. Kreutz, F.M. Ramos, and P. Verissimo, “Towards secure and dependable software-defined networks,” Proc. HotSDN’13, Hong Kong, China, Aug. 2013.
[7] S. Shin and G. Gu, “Attacking software-defined networks: a first feasibility study,” Proc. HotSDN’13, Hong Kong, China, Aug. 16, 2013.
[8] N.-N. Dao, J. Park, M. Park, and S. Cho, “A feasible method to combat against DDoS attack in SDN network,” Proc. International Conference on Information Networking (ICOIN), pp.309–311, 12-14 Jan. 2015.
[9] 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.
[10] J. Li, S. Berg, M. Zhang, P. Reifer, and T. Wei, “DrawBridge—software-defined DDoS resistant traffic engineering,” Proc. SIGCOMM’14, Chicago, IL, USA, Aug. 2014.
[11] S. Lim, J. Ha, H. Kim, Y. Kim, and S. Yang, “A SDN-Oriented DDoS blocking scheme for botnet-based attacks,” Proc. 6th Conference on Ubiquitous and Future Networks (ICUFN), pp.63–68, 8-11 July 2014.
[12] E.S. Page, “Continuous inspection scheme,” Biometrika, vol.41, no.1, pp.100–115, 1954.
[13] B.E. Brodsky and B.S. Darkhovsky, Nonparametric methods in change-point problems, Kluwer Academic Publishers, 1993.
[14] OpenFlow Switch Specification V1.4.0. https://www.opennetworking.org/images/stories/downloads/ondf-resources/onf-specifications/openflow/openflow-switch-v1.3.5.pdf.
[15] Tribe Flood Network 2000 (TFN2K), http://mixter.void.ru/papers.html.
[16] MIT Lincoln Laboratory, DARPA intrusion detection scenario specific datasets 2000, http://www.ll.mit.edi/IST/.
[17] Tcpreplay tools, http://tcpreplay.synin.net/.
[18] About POX, http://www.noxrepo.org/pox/about-pox.
[19] Open vSwitch, http://openvswitch.org/.
[20] R. Braga, E. Mota, and A. Passito, “Lightweight DDoS flooding attack detection using NOX/OpenFlow,” Proc. 35th Conf. on Local Computer Networks (LCN), Denver, pp.408–415, IEEE Press, 2010.
[21] D.K.Y. Yau, J.C.S. Lui, F. Liang, and Y. Yam, “Defending against distributed denial-of-service attacks with max-min fair server-centric router throttles,” IEEE/ACM Trans. Netw., vol.13, no.1, pp.29–42, 2005.
[22] K.J. Houle, G.M. Weaver, N. Long, and R. Thomas, “Trends in denial of service attack technology,” Technical Report, CERT and CERT Coordination Center, 2001.
[23] J. Jung, B. Krishnamurthy, and M. Rabinovich, “Flash crowds and denial of service attacks: Characterization and implications for CDNs and web sites,” Proc. International World Wide Web Conference, pp.252–262, May 2002.
[24] P. Tao, C. Leckie, and K. Ramamohanarao, “Protection from distributed denial of service attacks using history-based IP Filtering,” Proc. IEEE International Conference on ICC’03, Communications, 1(1): pp.482–486, 2003.
[25] A.N. Shiryaev, “The problem of the most rapid detection of a disturbance in a stationary process,” Soviet Math. Dokl, vol.2, pp.795–799, 1961.
[26] A.N. Shiryaev, “On optimum methods in quickest detection problems,” Theory of Probability and Its Applications, vol.8, no.1, pp.22–46, Jan. 1963.
[27] B.H. Bloom, “Space/Time trade-offs in hash coding with allowable errors,” Commun. ACM, vol.13, no.7, pp.422–426, 1970.
[28] H. Liu and M.S. Kim, “Real-time detection of stealthy DDoS attacks using time-series decomposition,” Proc. IEEE International Conference on ICC, pp.1–6, 23–27 May 2010.
[29] J. Kang, Z. Zhang, and J. Ju, “Protect e-commerce against DDoS attacks with improved D-WARD Detection System,” Proc. IEEE Conference on e-Technology, e-Commerce and e-Service, pp.100–105, 2005.
[30] D. Kotani and Y. Okabe, “A packet-in message filtering mechanism for protection of control plane in OpenFlow Networks,” Proc. 10th ACM/IEEE Symposium on Architecture for networking and Communication, pp.29–40, Oct. 2014.
[31] P. Tao, C. Leckie, and K. Ramamohanarao, “Proactively detecting distributed denial of service attacks using source IP address monitoring,” Lecture Notes in Computer Science, vol.3042, pp.771–782, 2004.
[32] S. Scott-Hayward, G. O’Callaghan, and S. Sezer, “SDN security: a survey,” Proc. IEEE SDN4FNS, 2013.