FlowTrApp: An SDN Based Architecture for DDoS Attack Detection and Mitigation in Data Centers

Chaitanya Buragohain, Nabajyoti Medhi
Dept. of CSE, NIT Meghalaya
Shillong, India
chaitanyanitm@gmail.com, nabajyoti.medhi@nitm.ac.in

Abstract— **Distributed Denial of Service attack (DDoS) is one of the severe security problems in data centers. In present times, data center operators adopt several hardware based dedicated measures for detection and mitigation of such attacks. It is a challenging task always to detect and mitigate DDoS attacks completely. Software Defined Network (SDN) provides a central control over the network which helps in getting the global view of the network. In this paper, we propose an SDN framework for data centers named FlowTrApp which performs DDoS detection and mitigation using some bounds on two per flow based traffic parameters i.e., flow rate and flow duration of a flow. It attempts to detect attack traffic ranging from low rate to high rate and long lived to short lived attacks using an SDN engine consisting of sFlow based flow analytics engine sFlow-RT and an OpenFlow controller. The proposed framework of FlowTrApp has been implemented in mininet emulator which outperforms an OpenFlow based QoS approach for DoS attack mitigation.**

Keywords— Distributed Denial of Service (DDoS); Software Defined Network (SDN); Data Center; sFlow-RT; OpenFlow;

I. Introduction

The extensive study of DDoS attacks started from more than two decades, but day by day they are increasing in number and becoming more complex which are negatively impacting a broad spectrum of internet applications [16] and many cloud based data centers. DDoS attacks can be of many different types. In particular, flooding based DDoS attacks, such as, ICMP, UDP and TCP SYN sends a huge number of fake packets to victim [6] which in turn depletes computing resources like CPU, memory and network bandwidth of the target network. As an example, ICMP and UDP flooding attacks consume the bandwidth of the victim, while in case of TCP SYN flooding; the attacker exhausts the connection table of the victim by sending SYN packets. On the other hand, detection of Layer 7 or application layer attack is very much complex. These Application level DDoS flooding attacks target specific application characteristics like DNS, HTTP etc. [16]. For example, a slow HTTP GET flood can be combined with UDP flood, which misleads the victim to UDP attack, while the HTTP flood slowly deplete the HTTP server resources.

SDN [20] provides a new network architecture in which the network control plane is decoupled from the data plane. This centralized control opens a number of opportunities for the network security problems. Although it brings numerous benefits by separating out the control from the data plan, there is a contradictory relationship exists between DDoS attacks and SDN [18]. This separation introduces new kinds of attack. An attacker can detect a software-defined network with the help of some basic SDN characteristics [8]. Since, for every new packet arriving at the ingress switch, the data plane asks the control plane for the new rule, the response time for already existing rule and the response time for installing the new rule can be differentiated. Based upon the response time, an attacker can easily differentiate a software-defined network from a traditional one. The centralized controller can be a potential target of the attackers. Whenever a new packet arrives at the ingress switch of the SDN network, it will check for a flow entry for that particular packet. Since the packet is new and the switch does not have any matching flow in the flow table, it will forward the packet to the controller. In this way, by continuously sending spoofed packets, attackers can overload the flow table in the switch as well as the bandwidth between the controller and switch can be occupied. As a result, the stability and quality of service of the network degrades and it may lead to shutdown of the whole network.

SDN standard like OpenFlow [21] is nowadays getting widely used. OpenFlow provides the facility of flow-match and action on the switches and routers. OpenFlow facilitates traffic engineering as well as security solutions for data centers and WAN. A central OpenFlow controller provides the global view of the network which provides better manageability but it also suffers the problem of lack of scalability due to congestion in the controller for a large network. SDN tools like sFlow [22] is a collector technology used for gathering flow statistics from the switches and routers which makes the network easily visible. It also enables the network admin to set some filters on the incoming or outgoing traffic and thereby it provides some security measures against the volumetric attacks.

In this paper, we propose a new framework for DDoS detection and mitigation using sFlow and OpenFlow. The proposed mechanism first matches an incoming flow with a legitimate sample of traffic and then installs mitigation actions if a flow is found not lying in the bounds of legitimate traffic pattern.

The remainder of the paper is organized as follows. Section II presents some reviews and related works. Section III introduces the proposed system architecture. Simulation results are discussed in section IV. Section V concludes the paper.

II. Related works

In this section, we discuss several notable works and investigations made in order to solve the DDoS problem with SDN and without SDN. Major functionalities of DDoS detection and mitigation schemes can be implemented using a centralized SDN controller. There are several works done on the security of SDN networks [8], but we are limited to the discussion of handling DDoS attack using SDN approach.

An SDN based cloud environment for DDoS attack detection is proposed in [2]. The module consists of two parts: (i) an anomaly-based attack detection module DaMask-D, and (ii) an attack mitigation module DaMask-M. The DaMask-D module is an anomaly detection module which is based on a graphical probabilistic inference model. The mechanism mainly focuses on analyzing the impact of DDoS attack in hybrid cloud.

An Orchestrator–based architecture is proposed in [3], which utilizes network monitoring and SDN control functions to develop security applications.

In [4], traffic flows are classified into four categories in terms of flow duration and bandwidth utilization. A minimum bandwidth threshold can be considered over a minimum observation interval to distinguish a flow that will belong to the one of the flow classes. As an example, a flow that exceeds the threshold within that observation interval can be considered as a long-lived large flow. To facilitate different use cases, these two parameters can be programmed in the switches/routers.

The connections made by the abnormal users and the number of packets per connection sent by these users are less as compared to that of normal or frequent users [5]. On the basis of these two parameters: the average number of connections and minimum number of packets per connection, the characteristics of a DDoS attack can be addressed.

Layer 7 or the application layer attacks are more complex to detect, because they consume the resources at a very low rate. To overcome this kind of attack a shark tank concept is proposed [7] that has full application capabilities. This shark tank is kept under detailed monitoring and whenever a suspicious traffic is detected it is redirected to the tank for further filtering.

Botnets also consume the specific application servers at a very low rate and make it difficult to detect by using the anomalous traffic statistics, because the traffic generated by the botnets looks same as the legitimate traffic. To overcome the attack caused by botnets, multiple CAPTCHA coded IP addresses can be used and the legitimate traffic can be redirected to one of the new IP address of the victim server [9]. Using this property, traffic from botnets can be separated out from the traffic of legitimate users.

During the DDoS attack, survival time of the whole network system can be increased by applying two level load balancing between the servers and between the network devices [10]. Layer 7 load balancing can be done by splitting the traffic streams between the endpoint servers and layer 4 load balancing can be done by splitting the packets through different paths in the network. This two level load balancing can be made independent from each other.

[11] uses entropy as a measure to detect DDoS attacks against SDN controller. Entropy is measured on the basis of the randomness of the incoming packets. For each new incoming packet entropy increases with respect to the randomness. But at the attack time, since excessive incoming packet comes from the same source, the entropy decreases with respect to the randomness of the packets. A particular entropy value can be used as a threshold with respect to which an attack is determined.

SDN based CDNi networks with multi-defense strategy to provide services against DDoS attacks have been studied in [12].

Neural network and biological danger theory can also be used for risk assessment in SDN to mitigate the DDoS attack [13]. Risk is calculated on every host and the report is send to an Alien Vault VM which monitors traffic. If the risk of the monitored traffic is above some predefined value then instructions are sent to the SDN controller to install some appropriate controls so that the SDN can enter in a proactive mode in which all controls are already installed in the switches. This proactive mode also reduces the delay caused by the flows sent to the controller for analysis.

Cloud computing provides a broad flexible network service to the users. But, DDoS attacks in cloud computing environment are becoming larger and more frequent. In [14], authors discuss about some defense mechanisms against DDoS attacks in cloud computing using SDN.

HTTP based cloud services can be protected from DDoS attacks by applying several multi-stage detection modules to detect attacks more precisely [15].

FRESCO [17] provides a security application development framework in the SDN controller. The modular libraries are the elementary processing units of FRESCO, which can be shared and linked together to provide complex network defense applications like security monitoring and threat detection.

Some of the security applications like SENSS [19] also include ISPs for on demand service to their customers for providing DDoS mitigation. It requires cooperation between the ISPs and customer who jointly works to mitigate the attack. ISPs and customer both may have their own SDN controllers [6] and the modules to detect DDoS runs in the customer network which generates the security alerts. Upon receiving the request from the customer, ISPs change the label of the suspicious traffic and redirect it to the security middle-boxes.

In [28], authors use a QoS based approach using OpenFlow queues to mitigate malicious traffic.

For detecting and mitigating anomalous traffic in SDN a combination of OpenFlow and sFlow mechanism is presented in [1] which is the closest to our work. As claimed by the authors, this mechanism tries to detect the attack patterns in real time. For collecting the data, sFlow is maintained in the edge switches and for mitigating rules are combined with the OpenFlow technology. Authors use three modules in their proposal [1]. First module is responsible for collecting required flow statistics. Second module is responsible for anomaly detection and identification which collects the harvested data set from the first module. Depending upon different network environments, the choice of anomaly detection algorithm can be varied. The third module is mainly responsible for attack mitigation which uses OpenFlow mechanism. It uses sFlow which avoids using of OpenFlow packet counter statistics.

In several works based on OpenFlow such as that mentioned in [10] are unaware of controller scalability issue with a growing rate of DDoS traffic. Putting complete load on a single OpenFlow controller may lower the controller performance. Some other works like [1], [4] use the similar approach like our model but the authors have not defined a clear duty division between the functionalities of central network controller technology like OpenFlow [21] and traffic statistics collector technology like sFlow [22]. The work done in [28] uses OpenFlow based QoS features like queuing of flows to limit the malicious traffic rate but it allows a significant portion of malicious traffic to pass through the network even after rate limitation.

In this paper, we come up with a novel approach to detect and classify DDoS attacks in a data center network and to mitigate them using a fruitful combination of sFlow and OpenFlow. Unlike the works mentioned earlier in this section, our proposed work provides the division of labor between sFlow and OpenFlow which not only improves the detection and mitigation process but also improves the SDN controller performances.

III. PROPOSED SYSTEM DESIGN

In this section, we discuss the FlowTrApp architecture for DDoS sensitive cloud and data-center applications. FlowTrApp does DDoS detection and mitigation based on an application specific L7 constraint set by the application admin i.e., there can be only a single application session possible for an HTTP request from a specific IP address. FlowTrApp attempts to mitigate DDoS attacks on web based applications with secure transactions e.g., railway reservation portal like IRCTC etc. IRCTC website nowadays allows an IP address to do a single transaction at a time. As an example, in an institute campus with a single public IP address, multiple users cannot book railway tickets at the same time. This constraint is applied in L7 as a pre-condition for the FlowTrApp algorithm. After setting the L7 constraint, a network administrator has to manually set some bounds on two per-flow based network parameters such as flow rate and flow duration to define a legitimate traffic pattern of the specific application i.e., traffic rate and traffic duration based on how much traffic a legitimate client do generally send and how longer.

FlowTrApp algorithm takes the legitimate bound information as the input and based on this information it classifies a traffic flow as either attack traffic or legitimate traffic. An initial manual survey is done by a set of legitimate users who perform valid operations using different kinds of devices on the application for legitimate durations of time. A traffic tuple is prepared by observing the minimum and maximum values of traffic rate and duration of the set of data obtained from the usage of legitimate users. This step must be done every time an operational change in application is done which may change the usage pattern of a legitimate user. From the traffic tuple, we can find a lower limit as well as an upper limit of traffic rate and duration. The threshold values of the traffic rate and flow duration are represented as a flow traffic tuple (FTT) which can be defined for any application in the following pattern:

$$<flow\_rate_{min}, flow\_rate_{max}, duration_{min}, duration_{max}>$$

Here, flow_rate defines the data rate of a flow and duration defines the duration or life-time of the flow. Based on the comparison with the flow traffic tuple values, an incoming flow is categorized either as a legitimate or an attack flow. An incoming flow having traffic rate and flow duration lying in the range of the FTT of an application is termed as the legitimate flow for that application.

Based on the FTT of an appliance, a malicious flow can fall into any of the significant attack categories as described below:

- A flow with higher traffic rate than the upper limit of the FTT but for duration less than the lower limit of traffic duration is termed as the High Rate Spike (HRS) attack flow
- A flow with traffic rate higher than the upper limit of the FTT but for duration less than the upper bound of traffic duration is termed as the Short Lived High Rate (SLHR) attack flow.
- A flow having traffic rate and traffic duration exceeding the upper bounds set by the appliance's FTT is termed as the Long Lived High Rate (LLHR) attack flow.
- A flow with traffic rate lying in the legitimate range of the FTT but for duration longer than the upper limit of traffic duration is termed as the Idle User (IS) attack flow.
- A flow with lower traffic rate than the lower limit of the FTT but for duration longer than the upper bound of traffic duration is termed as the Long Lived Low Rate (LLLR) attack flow..
- Characteristics of an attack flow not falling in any of the above categories cannot be detected in real time.

The proposed FlowTrApp architecture has the following major operational parts:

- Defining the thresholds of legitimate FTT from the traffic behavior of a group of legitimate users using a flow statistics collector like sFlow-RT [24],
- Identifying the malicious flows falling in any of the five attack categories mentioned earlier using sFlow-RT and passing the flows in low rate.
- Writing OpenFlow rules in the OpenFlow enabled switches to reactively block the malicious sources for a specific duration if the source is found to send attack traffic frequently.

The proposed algorithm of FlowTrApp can be used for both Layer 2 and Layer 3 addresses. In this paper, we are not discussing about the aggregation of multiple flows for distributed attack detection. We rather take the SDN approach for per-flow based DDoS attack detection and mitigation. According to our algorithm, detection and mitigation can be done on a per-flow basis by matching either IP or MAC addresses.

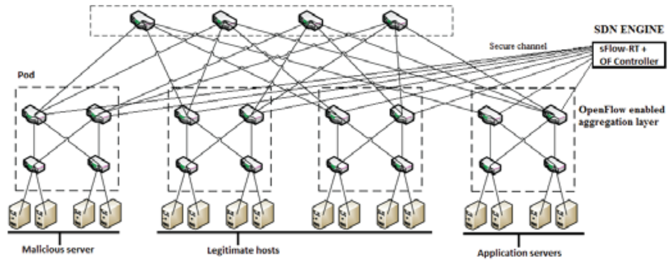

Fig. 1. A 4-port fattree topology with 16 hosts along with the proposed SDN engine.

In this paper, we limit our study within a data center network where some misbehaving tenants can generate DDoS traffic inside the network. FlowTrApp algorithm uses the FTT information of a specific application. We take Fattree [27] topology to portray a data center scenario. A common Fattree based DC network generally consists of three layers of switches i.e., core layer, aggregation layer and access layer. Core layer switches interface to the WAN router and access layer switches interface to the servers. In FlowTrApp architecture, we take an OpenFlow enabled aggregation layer where all the switches in the aggregation layer are OpenFlow enabled. An SDN engine is used monitor traffic in the aggregation layer. It consists of an sFlow-RT application with an OF controller. When new traffic flow arrives at the aggregation switches which interface the target application servers, the flow is sent to the sFlow-RT of SDN engine for monitoring. The OF controller keeps the global view of the network and installs the basic forwarding rules in the OpenFlow enabled switches. sFlow-RT checks the flow statistics on a per-flow basis. If the traffic characteristics fall in any of the attack categories then the flow is checked by the Mitigate algorithm which evaluates how many times the source address attempted to attack. If the source attempts to send the number of flows more than a random legitimate counter value violating the FTT then it is blocked for $duration_{max}$ time in order to ensure a safe congestion free network. Otherwise, a flow from such source address can be passed without blocking but in a rate $flow\_rate_{min}$ if $flow\_rate > flow\_rate_{min}$, otherwise in *flow_rate* to avoid congestion due to high rate traffic flows.

Algorithm FlowTrApp
Require: L7 constraint to allow a single request at a time to the application from one IP address i.e. an IP address cannot send multiple requests to the application in any duration of time. A Flow Traffic Tuple (FTT) is to be prepared based on the traffic behavior of a group of legitimate users.

Lets initialize a blank list *maliciousUser = { }* which contains source addresses violating FTT against their *malCounter* values initialized to 0. *malCounter* suggests how many times a particular source address behaves as malicious. Assign a random number as *legitimate_malCounter* value which suggests the upper limit of *malCounter* for a specific source address.

```
1: procedure FlowTrApp
2: for each new incoming flow at any router or switch do
3:     newUserList{} = Source IP/MAC of the new flow
       {IP address to be considered when detection is done in L3.
       Otherwise, MAC address can be considered when the
       traffic is coming from a tenant within the same data
       center}
4:     Install OpenFlow rules for forwarding the new
       IP/MAC in switch flow table
5:     Monitor the rate and duration of the flow using sFlow-RT
6:     if flow_rate >= $flow\_rate_{max}$ and flow duration <=
           $duration_{min}$ of FTT then {HRS attack}
7:             Mitigate(source_address, $flow\_rate_{min}$)
8:     else if flow_rate >= $flow\_rate_{max}$ and flow duration
           <= $duration_{max}$ of FTT then {SLHR attack}
9:             Mitigate(source_address, $flow\_rate_{min}$)
10:    else if flow_rate >= $flow\_rate_{max}$ and flow duration >=
           $duration_{max}$ of FTT then {LLHR attack}
11:            Mitigate(source_address, $flow\_rate_{min}$)
12:    else if $flow\_rate_{min}$ < flow_rate < $flow\_rate_{max}$ and flow
           duration >= $duration_{max}$ of FTT then {IS attack}
13:            Mitigate(source_address, $flow\_rate_{min}$)
14:    else if flow_rate <= $flow\_rate_{min}$ and flow duration
           >= $duration_{max}$ of FTT then {LLLR attack}
15:            Mitigate(source_address, flow_rate)
16:    else
17:            Attack cannot be detected. Let the flow pass
               in original flow_rate.
18:    end if
19: end for
20: end procedure
```

```
Function Mitigate(source_address, flow_rate)
1: If the source address of the incoming flow exists in the
   maliciousUser list
2:     do malCounter++
3:     if malCounter > legitimate_malCounter,
4:         Set OpenFlow rule hardTimeout = $duration_{max}$ to block
           the source address for $duration_{max}$ time.
5:         Remove the source from newUserList
6:     else
7:         Send the flow in flow_rate
```

Scalability of the algorithm depends on the performance of the OF controller and in this paper, we try to minimize the load on OF controller by using sFlow for switch statistics monitoring and keeping a global controller for normal forwarding decisions.

The above algorithm accomplishes the three major operational parts of the architecture as stated in an earlier section. By following a few steps, it can be installed in the network of a target application server:

1. Manually collect the legitimate traffic pattern from a group of legitimate users.

2. Prepare the Flow Traffic Tuple using the threshold values of traffic rate and duration obtained in step 1.
3. Install hybrid OpenFlow switches in the core and access layer of the network including the edge router if it is OpenFlow enabled.
4. Run a flow statistics collector such as sFlow-RT along with OpenFlow controllers kept in a controller device and connect them to the OpenFlow switches.
5. Run the FlowTrApp algorithm script in the controller device.

IV. Simulation results

The proposed architecture of FlowTrApp is tested using Mininet emulator [23]. The algorithm is implemented using inMon sFlow-RT [24] and Floodlight controller [25]. sFlow-RT is used as a collector of flow statistics which can automatically detect large flows in real time. It also provides the separate per-filters to detect different types of attacks. In order to install on demand flow entries using REST API in the switches and also to install flows reactively based on the algorithm, Floodlight controller is used.

A. *Experiment 1:*

We create sample data center topology with a radix-4 Fattree [27] topology having 16 hosts and 20 switches inside Mininet. As shown in fig. 1, 4 hosts in the rightmost pod are application servers, leftmost server is the malicious host and the remaining hosts are taken as legitimate hosts. sFlow-RT and OF controller are connected to the aggregation switches in pod 4. Each host to switch link is kept at a capacity of 100Mbps and each switch to switch link is kept at 400Mbps each. We generated concurrent flows from all the hosts using Iperf [26] traffic generator targeted to the 4 application servers.

1) *Testing scenario:*

In this topology, UDP flood traffic in the range of 10Mbps to 400Mbps was sent to the 16 hosts for durations ranging from 1 second to 100 seconds using Iperf [26]. A sample FTT for the application servers was set to [$flow\_rate_{min}$=*50Mbps*,$flow\_rate_{max}$=*100Mbps*, $duration_{min}$=*5secs*, $duration_{max}$=*10secs*] in sFlow-RT.

2) *Result:*

- UDP flood attack was detected at around every 10 seconds.
- Traffic rate was limited in the range of 60 to 95 Mbps for all the combinations of traffic rate and duration. The range of the traffic rate was found to be restricted around the upper bound set for the FTT.

We repeated the experiment by taking different FTTs and each time we achieve traffic rate within the FTT range by using FlowTrApp algorithm. The simulated upper bound for the traffic rate is obtained for each pair of traffic rate and duration is found to be approximately equal to the theoretical upper bound.

B. *Experiment 2:*

In this part, we compare the performance of FlowTrApp algorithm against a QoS based mechanism as given in [28]. In [28], enqueue operation is used for detected flow entries. In the same Fattree topology used earlier, we use enqueue operation on the two aggregation switches in the pod where application servers are located. Enqueue action is installed with same lower and upper bound on the flow rate as taken in the FTT selected in the previous experiment. Duration of flow is kept at 20 seconds. To have a fair comparison, we implement FlowTrApp algorithm on the same two aggregation switches as earlier. We send 100Mbps TCP traffic from a malicious host to one of the application servers in both the cases.

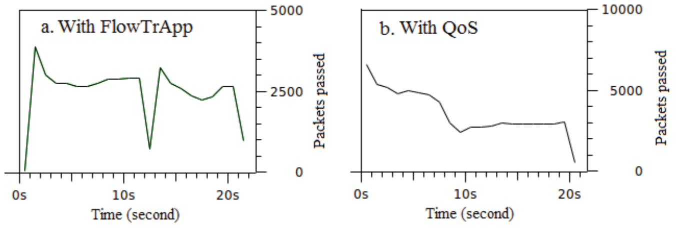

Fig. 2. Comparison between a. FlowTrApp and, b. QoS based approach

Fig. 2a, 2b shows that QoS approach allows more malicious packets to pass than FlowTrApp. FlowTrApp allows 2500 packets on an average whereas QoS approach allows more than 5000 packets on an average to pass in the first 10 seconds.

C. *Experiment 3:*

In this section, we test the efficiency of FlowTrApp in terms of OpenFlow traffic. We compare the performance of our algorithm with a load-balancing based DDoS mitigation mechanism similar to that mentioned in [10].

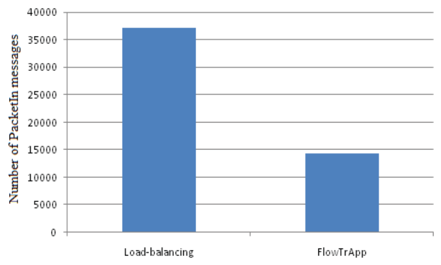

Fig. 3. Number of PacketIn messages used in load-balancing mechanism and in FlowTrApp

We use the same topology as used in experiment 1 to test the performances. We generate ICMP flood from the malicious hosts towards the application servers. With FlowTrApp, less number of PacketIn messages pass through the network as compared to that in case of the load-balancing approach [Fig. 3]. FlowTrApp minimizes the OpenFlow PacketIn traffic which may be generated due to attack traffic for two reasons. First, the interaction of only aggregation switches with the SDN engine. Second, sFlow takes the responsibility of flow statistics analysis which reduces the duty of the OpenFlow controller.

V. Summary and Future Work

In this paper, we proposed and tested the proposed architecture of FlowTrApp. It provides SDN based architecture for detecting and mitigating high rate as well as low rate DDoS attacks in data centers. The proposed algorithm can categorize any incoming attack traffic flow by matching with an application specific legitimate flow traffic tuple. Rather than blocking an address in the first attempt, mitigation is done when a malicious user is found to send attack traffic frequently. SDN technologies like OpenFlow and flow statistics collector technology like sFlow are used to implement the architecture. Use of both these technologies enables the duty sharing between the sFlow-RT application and the OpenFlow controller towards DDoS attack detection and mitigation which enhances the performance of FlowTrApp. After comparison, our model is found to perform better than an existing QoS based approach. As a future work, attempts will be made for expanding and generalizing our algorithm to handle different other types of DDoS attacks.

References

[1] 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," Journal of Computer Networks, vol. 62, pp. 122-136, April 2014.

[2] B. Wang, Y. Zheng, W. Lou and, Y. T. Huo," DDoS attack protection in the era of cloud computing and Software-Defined Networking," Journal of Computer Network, vol. 81, pp. 308-319, April 2015.

[3] A. Zaalouk, R. Khonodoker, R. Marx, and K. Bayarou," OrchSec: An Orchestrator-Based Architecture For Enhancing Network-Security Using Network Monitoring And SDN Control Functions", 2014 IEEE Network Operations and Management Symposium (NOMS), pp. 1-9, May 2014.

[4] R. Krishnan, D. Krishnaswamy, and D. Mcdysan, "Behavioral Security Threat Detection Strategies for Data Center Switches and Routers," 34th IEEE conference on Distributed Computing Systems Workshops (ICDCSW), pp. 82-87,July 2014

[5] D. Nhu-Ngoc, J. Park, M. Park, and S. Cho, "A Feasible Method to combat against DDoS Attack in SDN Network," International Conference on Information Networking (ICOIN), pp. 309-311, January 2015

[6] R. Sahay, G. Blanc, Z. Zhang, and H. Debar, "Towards Autonomic DDoS Mitigation using Software Defined Networking," NDSS Workshop on Security of Emerging Networking Technologies (SENT 2015) (2015).

[7] M. Shtern, R. Sandel, M. Litoiu, C. Bachalo, and V. Theodorou, "Towards mitigation of low and slow application ddos attacks," IEEE International Conference on Cloud Engineering (IC2E), pp. 604-609, March 2014.

[8] S. Shin, and G. Gu, "Attacking software-defined networks: A first feasibility study," Proceedings of the second ACM SIGCOMM workshop on Hot topics in software defined networking (HotSDN'13), pp. 165-166, 2013.

[9] S. Lim, J. Ha, H. Kim, Y. Kim, and S. Yang "A SDN-oriented DDoS blocking scheme for botnet-based attacks," Sixth International Conf on Ubiquitous and Future Networks (ICUFN), pp. 63-68, July 2014.

[10] M. Belyaev, and S. Gaivoronski, "Towards load balancing in SDN-networks during DDoS-attacks," First International Science and Technology Conference (Modern Networking Technologies)(MoNeTeC), pp. 1-6, October 2014.

[11] S. M. Mousavi, and Marc St-Hilaire, "Early detection of DDoS attacks against SDN controllers," International Conference on Computing, Networking and Communications (ICNC), pp. 77-81, February 2015.

[12] N. I. Mowla, D. Inshil Doh, and K. Chae, "Multi-defense Mechanism against DDoS in SDN Based CDNi." Eighth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), pp. 447-451, July 2014.

[13] I. Mihai-Gabriel, and P. Victor-Valeriu, "Achieving DDoS resiliency in a software defined network by intelligent risk assessment based on neural networks and danger theory," 15th International Symposium on Computational Intelligence and Informatics (CINTI), pp. 319-324, November 2014.

[14] Q. Yan, and F. Yu, "Distributed denial of service attacks in software-defined networking with cloud computing," IEEE Communications Magazine, Issue 4, vol. 53, pp. 52-59, April 2015.

[15] V. S. Huang, R. Huang, and M. Chiang, "A DDoS Mitigation System with Multi-stage Detection and Text-Based Turing Testing in Cloud Computing," 27th International Conference on Advanced Information Networking and Applications Workshops (WAINA), pp. 655-662, March 2013.

[16] 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, Issue 4, vol. 15, pp. 2046-2069, March 2013.

[17] S. Seungwon, P. Porras, V. Yegneswaran, M. Fong, G. Gu, and M. Tyson, "FRESCO: Modular Composable Security Services for Software-Defined Networks," ISOC Network and Distributed System Security Symposium, February 2013.

[18] S. Scott-Hayward, G. O'Callaghan, and S. Sezer, "Sdn security: A survey," . IEEE SDN for Future Networks and Services (SDN4FNS), pp. 1-7, November 2013.

[19] A. Alwabel, M. Yu, Y. Zhang, and J. Mirkovic, "SENSS: observe and control your own traffic in the internet," ACM SIGCOMM Computer Communication Review, Issue 4, vol. 44, pp. 349-350, October 2014.

[20] "SDN Definition," https://www.opennetworking.org/sdn-resources/sdn-definition.

[21] "OpenFlow Definition," https://www.opennetworking.org/sdn-resources/openflow.

[22] "Benefits of Flow Analysis Using sFlow: Network Visibility, Security and Integrity," A Lancope technical white paper, http://www.gisec.ae/files/lancope_generic_sflow_wp.pdf.

[23] "Mininet Emulator," http://mininet.org/

[24] "inMon sFlow-RT," http://www.inmon.com/products/sFlow-RT.php.

[25] "Floodlight Controller," http://www.projectfloodlight.org/floodlight/.

[26] "Iperf-The network bandwidth measurement tool," https://iperf.fr/.

[27] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture", in Proc. ACM SIGCOMM '08, (Seattle, USA), pp. 63-74, Aug. 17-22, 2008.

[28] Y. E. Octian, S. Lee, and H. Lee, "Mitigating Denial of Service (DoS) Attacks in OpenFlow Networks", in Proc. International Conference on Information and Communication Technology Convergence (ICTC), (Busan), pp. 325-330, Oct 2014.