Scientific Journal of Impact Factor (SJIF): 5.71

e-ISSN (O): 2348-4470
p-ISSN (P): 2348-6406

International Journal of Advance Engineering and Research Development

Volume 5, Issue 02, February -2018

Mitigation of controller and network attacks in software defined networking

M Jalasri[1], R S Yamuna[2], T Swetha[3]

[1]*Assistant Professor, Jeppiaar Maamallan Engineering College*
[2]*Student, Jeppiaar Maamallan Engineering College,*
[3]*Student, Jeppiaar Maamallan Engineering College*

Abstract- *Software Defined Networking, a recently proposed networking paradigm which has an intention of simplifying the management and maintenance of networking infrastructures. Under certain traffic, the required communication between the control and data plane can result in a bottleneck. An attacker can exploit this limitation to mount a new, network-wide, type of denial of service (DOS) attack, known as the control plane saturation attack and other sorts of networking attacks. The data plane, which consists of switches and routers, is responsible only for forwarding traffic, whereas control logic and functionality are moved to an external entity known as the SDN controller. The network intelligence is logically centralized in trusted software-based SDN controllers that provide an abstract view of underlying network resources. The abstraction of the flow broadly unifies the behavior of different SDN agents. Efficient and effective solution to mitigate the control plane and network attacks. In this paper, we have proposed a heuristic solution to approach its exact solution. Propose additional modifications to Line Switch, which allows for a more refined use of proxying based on incoming packets rate at the controller level, and address some of the possible attacks on legit clients. While reducing the introduced timing overhead, when compared to the current state. Through extensive simulations, it demonstrates that our heuristic algorithm has a good performance; that is, on average it can save about 50% of the total power consumption in the full SDN, having a distance less than 5% of the exact solution's power consumption. Moreover, it also achieves good performance in the partially deployed SDN, on average saving about 40% of the total power consumption when there are about 60% SDN nodes in the network.*

Keywords *- Software Defined Networking controller, SYN flood attack, Data Plane, Performance, Efficiency, Traffic Reduction, Attack Mitigation.*

I. INTRODUCTION

In the recent times, everyone is switching to Software Defined Networking (SDN) which has grasped the attention of more researchers. As SDN offers interfaces to implement fine-grained network management, monitoring, and control, it is considered a key element to network optimization algorithms [1].The SDN controller who plays an important role gets connected with all the nodes and switch thereby identifying knowing all the nodes and the possible paths that can optimally route to the destination from the sender node. Such programmability in the data plane where the SDN controller resides is widely regarded as expensive compared to simple forwarding protocols, such as Ethernet for which one could easily design purpose-built ASICs, such as Ethernet chips [2].The attacker can inject more fake packets to the destination thereby leading to traffic in the communication path within the authenticated sender and the server. The most popular emulator for SDN emulator is Mininet which is widely used in SDN experimentations and has several plug-in and add-ins and one such plug-in is Mininet-WIFI that enhances the Mininet environment with means for rapid prototyping and experimental evaluation of SDN for wireless environments [4].Almost all researcher and industry-wide staffs and the existing instantiations of the SDN paradigm use Open Flow which is the most widely adopted. Open Flow defines the standard API for communication between the data plane and the control plane, which identify the network traffic and the idea of flow tables. The control plane can program the flow tables of the Open Flow switches either proactively or reactively.

Each time an Open Flow switch receives an inbound flow for which it has no matching rule, it will request the installation of a new rule to the control plane. OpenFlow networks lack scalability between the data and control planes and this instigates the enabled targeted attacks by an external entity who crafts in the inbound stream of flow requests to inundate communications between the controller and switch in an adversary model [3].We propose additional modifications to the existing terms without the help of OpenFlow. Only with the help of the identified MAC information the whole stream or the link that has been established gets blocked up, therefore reduces the traffic and also the timing overhead.

II. PROBLEM STATEMENT

The existing instantiations of the SDN paradigm, Open Flow is the most widely adopted. Open Flow defines the standard API for communication between the data plane and the control plane, which identify the network traffic and the idea of flow tables. The control plane can program the flow tables of the Open Flow switches either proactively or reactively. Each time an Open Flow switch receives an inbound flow for which it has no matching rule, it will request the

installation of a new rule to the control plane. The existing networking paradigm suffers from a serious distortion where the data sent from a sender is sent back again to the sender if the path gets deviated or violations if any there exists only way to block the IP address which is being masked by a fake address by the attacker or the intruder. Thus blocking the IP address alone is not sufficient to prevent the attackers from gaining access. Single relay protocols the only single connection for each node connected within the network such as RT protocols.

III. SYSTEM DESIGN

There are two planes such as control plane and the data plane. The data plane, which consists of switches and routers, is responsible only for forwarding traffic .The data plane consists of flow tables along with the required software and hardware. It also contains the flow table where new rules are defined when there are no existing matching rules found in it. The control logic and functionality are moved to an external entity known as the SDN controller. Individual app agents are held within each individual application. The controller core exists in between the northbound application programming interface and the southbound application programming interface. The SDN controller gets linked up with the agent manager and the channel agent along with the network hub .The network hub in turn gets connected with the switches and the host. The SDN controller provides alternate and optimized routing information to the switch when the traffics are found and the dropping of packets are identified.

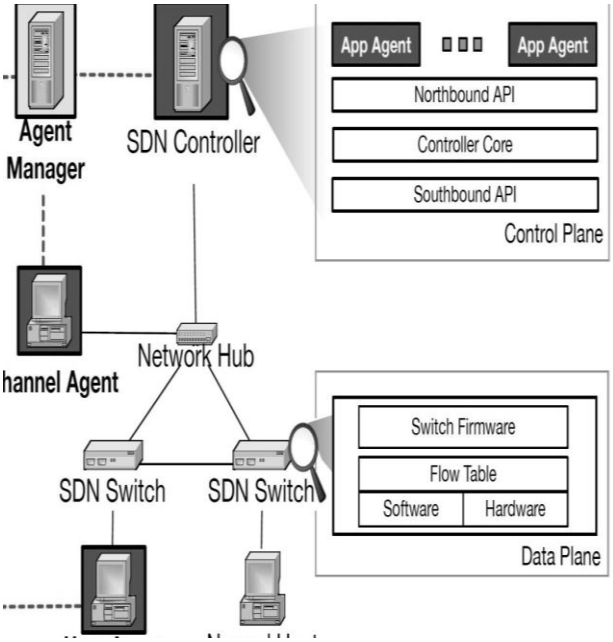

Fig 1: Architecture diagram

IV. PROPOSED WORK

A. Node Creation

Characterize an exceptionally straightforward topology with two nodes that are associated by a connection. A new node object is created with the command '$ns node' the simulator object will connect the nodes with a duplex link with the bandwidth, a delay of and a Drop-Tail queue. Next is to send some data from node to another node by creating an agent object that sends data from node to other agent objects that receive the data on the node. A constant bit rate [CBR] traffic generator is attached to set packet size, time interval. Create a Null agent which acts as traffic sink and attach it to the node. Tell the CBR agent when to send data and when to stop sending. The simulator object should give time interval for the simulation to execute the 'finish' procedure.

B. Drop Tail Queue

It is a simple queue management algorithms utilized by organizing schedulers in arrange gear to choose when to drop parcels. With tail drop, when the line is completely filled, the recently arriving bundles are dropped until the point that the line has enough space to acknowledge activity.

C. Constant Bit Rate (CBR)

It is useful for streaming multimedia content on constrained limit channels since it is the most extreme piece rate that issues not the normal, so CBR would be utilized to take the upside of the majority of the limit.

D. SDN Network Scenario

In this module, create the normal nodes (Device) and Controller on the wired network. Nodes are thus generated with agent object and CBR.SDN controller gets connected to all the nodes within the network and gets coped up with the switch. Perform the normal node operation on the node. Identify and transfer the data to the node in the wired network. Data are transferred between the nodes with help of agent object in each node. Identify the path of data flows on the wired network. The data flows in an optimized way where the shortest path is identified in advance.

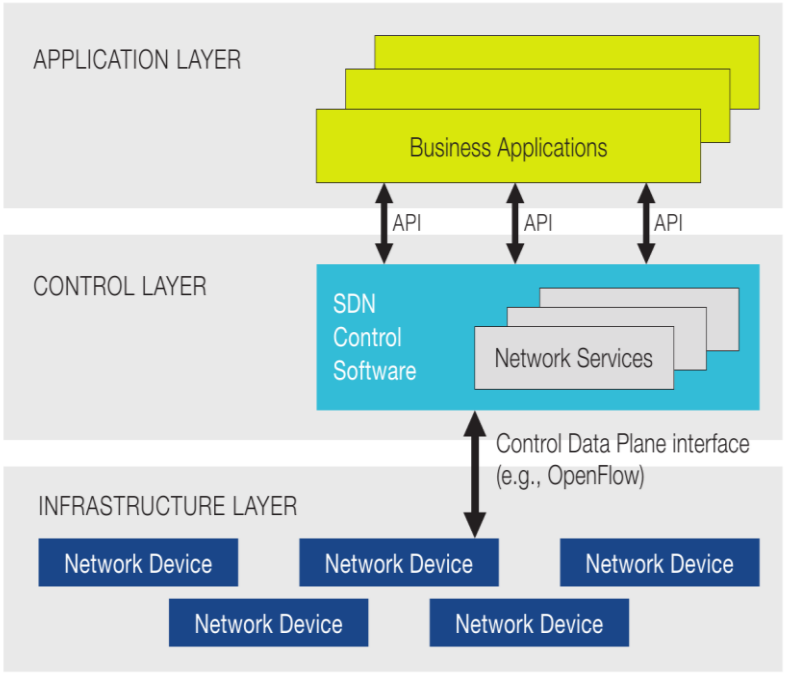

Fig 2: SDN network scenario

E. DDOS Attack Creation in SDN Network

The Attack has been created in the network. The attack gets into the network through a node. The main aim of the attack is to establish the connection with the server which is held up with another authenticated sender and to send more and more duplicated data packets thereby causing traffic in the network. The multistage attack is created at node 14 and node 11, can't able to control the device resulting in the dropping of queued packets which is from the authenticated sender. The sender without this knowledge keeps on constantly the sending the data to the sender. Detect multistage attacks solely. The analysis of attacks should take place. The loss in energy hints that the attacker has been exploiting the limitations in the network.

F. SYN flood Attack

It is a kind of Dispersed Disavowal of Administration (DDoS) assault that adventures some portion of the ordinary TCP three-path handshake to devour assets on the focused on server and render it unresponsive. Essentially, with SYN surge DDoS, the guilty party sends TCP association asks for quicker than the focus on the machine can process them, causing system immersion.

At the point when a customer and server build up an ordinary TCP, the Customer asks for association by sending SYN (synchronize) message to the server. The server recognizes by sending SYN-ACK (synchronize-recognize) message back to the client. Client reacts with an ACK (recognize) message, and the association is built up.

In an SYN surge assault, the assailant sends rehashed SYN parcels to each port on the focused on the server, frequently utilizing a phony IP address. The server, unconscious of the assault, gets different, clearly genuine solicitations to set up correspondence. It reacts to each endeavor with an SYN-ACK bundle from each open port. The vindictive customer either does not send the normal ACK or if the IP address is the caricature never gets the SYN-ACK in any case. In any case, the server under assault will sit tight for affirmation of its SYN-ACK parcel for quite a while.

At this time, the server can't shut down the association by sending an RST bundle, and the association remains open. Prior to the association can time out, another SYN parcel will arrive. This leaves an undeniably huge number of association's half-open and for sure SYN surge assaults are likewise alluded to as "half-open" assaults. In the long run, as the server's association flood tables fill, administration to true blue customers will be denied, and the server may even glitch or crash.

While the "work of art" SYN surge depicted above tries to debilitate organize ports, SYN parcels can likewise be utilized as a part of DDoS assaults that endeavor to stop up your funnels with counterfeit bundles to accomplish arrange immersion. The kind of bundle isn't critical. In any case, SYN parcels are frequently utilized in light of the fact that they are the most drastically averse to be dismissed naturally.

G. Attack Mitigation in SDN Network and performance Analysis

Create spanning trees according to traffic demand and current network topology. Allocate traffic in the spanning trees. If some link's loads exceed its capacity. Attack Mitigation efficient network management is based on a development of heuristic algorithm. Heuristics are important in practice because efficiency is often a high priority. Attack gets prevented with the help of MAC information. OSPF protocol helps in providing efficient communication between the nodes and the SDN controller along with the switch that gets linked up to the network. The whole stream that has raised by establishing the connection with server gets disconnected or blocked. Even though masking the machine with different many numbers of IP addresses cannot inject traffic when the stream gets blocked.

V. RESULT

A. Modified Heuristic Algorithm based on QOS (Quality Of Service)

Algorithm

Step 1: Create spanning trees according to traffic demand and current network topology, then go to step 2.

Step 2: Allocate traffic in the spanning trees. If some link's loads exceed its capacity, go to step 3, or if all traffic is allocated, we stop.

Step 3: Remove the overloaded links from the network topology, and use left traffic and new network topology to repeat step 1

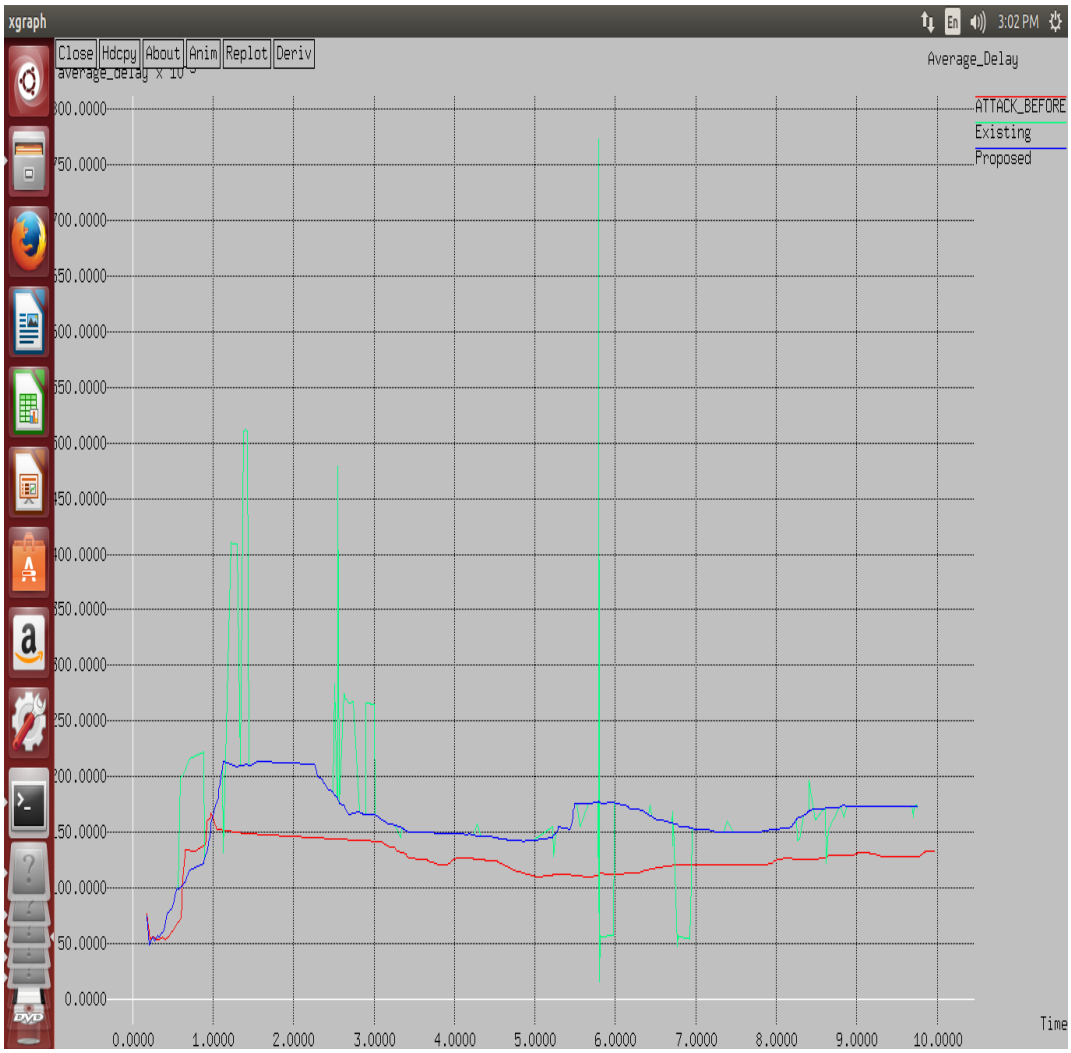

Fig. 3Average delay

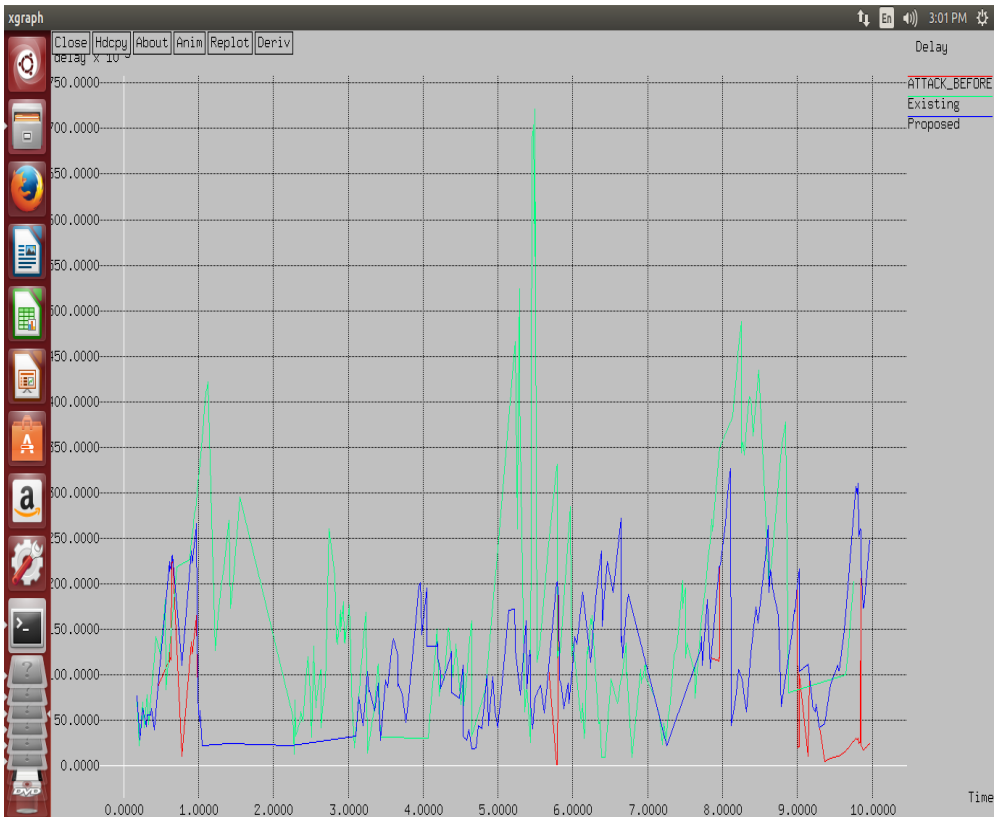

Fig. 4Delay

The Quality of Service (QoS) requirement of a point–to–point connection is typically specified as a set of constraints, which can be linked constraints or path constraints. A link constraint, such as the bandwidth constraint, specifies the restriction on the use of links. For example, the bandwidth constraint requires that each link along the path must be able to support certain bandwidth. A path constraint, such as the delay constraint, specifies the end–to–end QoS requirement for the entire path.

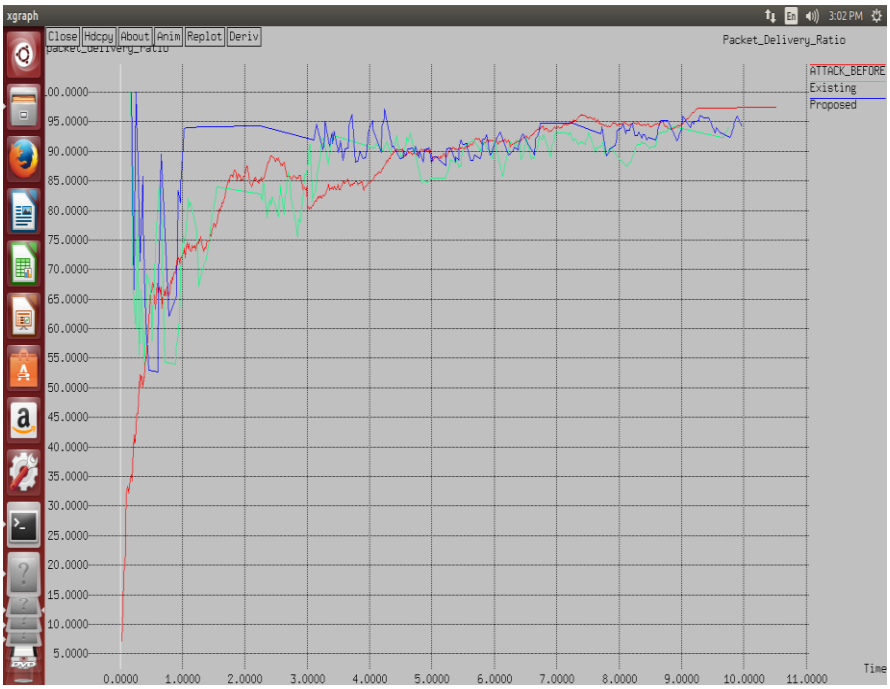

Fig. 5 Packet delivery ratio

Energy efficient network management is based on the development of heuristic algorithm. Heuristics are important in practice because efficiency is often a high priority. An efficient heuristic algorithm is the one which determines a solution within a reasonable time using reasonable resources.

For the types of problems considered in this work, a typical reasonable time frame is a few hours and a typical reasonable resource is a high-end personal computer (server). Our heuristic approach has been spatially tight to the problem (mathematical) model, we focused on energy consumption minimization in SDN, and the aim of heuristics is to build up a solution that offers the lowest energy consumption of the network in each time period.

During this process, heuristics must take into account different constraints. The proposed heuristic algorithm is composed of the two phases. In the first one is build up a feasible solution, local search (LS) starts with an initial solution and iteratively moves to the best candidate within the current neighborhood until no further improvement can be achieved.

The term heuristic is utilized for an algorithm which discovers arrangements among every conceivable one, however, they don't ensure that the best will be found, in this manner they might be considered as around and not exact algorithms. These algorithms generally discover an answer near the best one and they find it quick and easy. Sometimes these algorithms can be precise, that is they really locate the best solution, yet the algorithm is still called heuristic until the point when this best solution is ended up being the best. The strategy utilized from a heuristic algorithm is one of the known methods, such as greediness. For example, insatiability yet with a particular true objective to be straightforward and fast the figuring ignores or even smothers a segment of the issue's solicitations.

VI. CONCLUSION

Software-Defined Networking is foreseen as a means of using more efficient network resources and dynamically adapting the routing configuration over time. In this context, also show that the optimality gap does not diverge and we propose a control framework working on top of any routing solver to bind the network reconfiguration rate. Within this framework, it proposes a control policy that achieves the best trade-off between reconfiguration rate and optimality gap. The SDN Controller gives an essential issue of control to disperse security and arrangement data reliably all through the venture. We tentatively showed that it forces an unimportant overhead, which can be powerfully changed in accordance with fit the system needs, while effectively shielding the OpenFlow change and controller from assaults that can potentially disrupt the functionality of the network. Bringing together security control into one substance, similar to the SDN Controller, has the weakness of making an essential issue of assault; however, SDN can viably be utilized to oversee security all through the venture on the off chance that it is executed safely and legitimately.

REFERENCES

[1] Niels L.M.Van Adrichem, Christian Doerr and Fernando A.Kuipers, “Open NetMon:Network Monitoring in Open Flow Software-Defined Networks”, IEEE 2014.
[2] Gergely Pongracz, laszlo Molnar and Zoltan Lajos kis,”Removing Roadblocks from SDN: OpenFlow Software Switch Performance on Intel DPDK” IEEE 2013,pp. 62-67.
[3] Seungwon Shint,Vinod Yegneswaran, Philip Porras and Guofei Gut,VANT-GUARD” Scalable and Vigilant Switch Flow Management in Software-Defined Networks”, ACM 2013.
[4] Alexandru Stancu,Alexandru Vulpe,Octavian Fratu and Simona Halunga,”Wireless Transport emulator Based on LINC Software Switch”,Springer 2017.
[5] Piotr Rygielski, Marian Seliuchenko, Samuel Kounev and Mykhailo Klymash,”Performance Analysis of SDN Switches with Hardware and Software Flow Tables”,IEEE 2016.
[6] T. Peng, C. Leckie, and K. Ramamohanarao, “Survey of network-based defense mechanisms countering the DoS and DDoS problems,” *ACM Comput. Surv.*, vol. 39, no. 1, Apr. 2007, Art. no. 3.
[7] R. Kloti, V. Kotronis, and P. Smith, “OpenFlow: A security analysis,” in *Proc. IEEE ICNP*, Oct. 2013, pp. 1–6.
[8] K. Benton, L. J. Camp, and C. Small, “OpenFlow vulnerability assessment,” in *Proc. HotSDN*, 2013, pp. 151–152.
[9] M. Ambrosin, M. Conti, F. De Gaspari, and R. Poovendran, “LineSwitch: Efficiently managing switch flow in software-defined networking while effectively tackling DoS attacks,” in *Proc. ACM ASIA CCS*, 2015, pp. 639–644.
[10] D. Kreutz, F. M. Ramos, and P. Verissimo, “Towards secure and dependable software-defined networks,” in *Proc. HotSDN*, 2013, pp. 55–60.
[11] S. A. Mehdi, J. Khalid, and S. A. Khayam, “Revisiting traffic anomalydetection using software defined networking,” in *Proc. RAID*, 2011, pp. 161–180.
[12] J. Li, S. Berg, M. Zhang, P. Reiher, and T. Wei, “DrawBridge: SoftwaredefinedDDoS-resistant traffic engineering,” in *Proc. ACM SIGCOMM*, 2014, pp. 591–592.
[13] R. Braga, E. Mota, and A. Passito, “Lightweight DDoS flooding attack detection using NOX/OpenFlow,” in *Proc. IEEE LCN*, Oct. 2010, pp. 408–415.
[14] *NOX Network Control Platform*, accessed on Nov. 17, 2016. [Online].Available: http://www.noxrepo.org/nox/about-nox/
[15] Wang, Y. Guo, F. Hao, T. V. Lakshman, and S. Chen, “Scotch: Elastically scaling up SDN control-plane using vSwitch based overlay,” in *Proc. ACM CoNEXT*, 2014, pp. 403–414.

