Secured-OFS: A Novel OpenFlow Switch Architecture with Integrated Security Functions

Bao Ho(✉), Quoc Nguyen, Cuong Pham-Quoc, and Tran Ngoc Thinh

Ho Chi Minh City University of Technology, Vietnam National University - HCMC, Ho Chi Minh City, Vietnam
{7140219,51102795,cuongpham,tnthinh}@hcmut.edu.vn

Abstract. Although OpenFlow network protocol is a promising network approach with many advantages compared to traditional network approaches, it still suffers from network attacks. In this paper, we propose a novel architecture for an OpenFlow-based switch with associated multiple network security techniques, so-called Secured-OFS. The proposed Secured-OFS can not only function as a switch following the OpenFlow protocol but also help protect a network against many attack types. We implement the first FPGA-based prototype version of our proposed Secured-OFS using a Xilinx Virtex 5 xc5vtx240t device. In this first prototype version, we integrate two different DDoS defense techniques, Hop-Count Filtering and Port Ingress/Egress Filtering. The experimental results show that the switch not only fulfills the OpenFlow protocol but also be able to defense against DDoS attacks. The system achieves a maximum throughput at 19.729 Gbps while a 100 % DDoS attack detection rate is obtained.

Keywords: Software defined networking · OpenFlow network · Network security

1 Introduction

In the last decades, SDN [1] has been considered as a promising paradigm to manage and configure computer networks through a high-level abstraction. Compared to the traditional approach where computer networks are configured manually, the SDN approach has many benefits such as centralization control and monitoring, simple hardware devices, and high virtualization. The SDN architecture decouples network control from forwarding functions so that network control becomes programmable. The network control includes *controllers* programmed by network administrators through software interfaces. Each controller is responsible for handling several *forwarding devices* behaving forwarding functions. Those forwarding devices route network packets from a source node to a particular destination node according to network configuration.

As the most well-known instance of SDN, OpenFlow [2] is not only a quite popular implementation in academia but also an industry standard of SDN [3].

M. Akagi et al. (eds.), *Advances in Information and Communication Technology*,
Advances in Intelligent Systems and Computing 538, DOI 10.1007/978-3-319-49073-1_57

Based on the architecture of SDN, the OpenFlow network architecture also decouples network control function from the forwarding functions. Therefore, the OpenFlow network takes all the advantages of the SDN paradigm. Moreover, by optimizing elements such as controllers and forwarding devices, the OpenFlow network can be implemented as a software program or used as hardware platforms.

However, many security issues exist in both the SDN and OpenFlow network architectures. Research in [4] presents seven different threats in a SDN network which attackers can exploit to attack the network. OpenFlow networks also have some security threats that should be considered carefully. In the literature, there are some proposals to defend against possible attacks [5–7]. However, these approaches are only deployed as software programs. Moreover, research in the literature mainly focuses on optimizing controllers in OpenFlow networks [8,9]. There are still many open issues with forwarding devices. With the rapid progress in network services and network speed, high-performance and secure forwarding devices in OpenFlows networks is an essential demand.

With the fast increasing in the number of network attacks, hardware-based network defense plays an important role of a successful cyber-security strategy. In this paper, we propose a Secured-OpenFlow switch (Secured-OFS) with associated security functions. These Secured-OFSes can operate as forwarding devices in OpenFlow networks. We implement the first prototype version of Secured-OFS on the NetFGPA-10G [10] board which contains a Xilinx Virtex 5 xc5vtx240t FPGA device. The experimental results show that the system achieves high accuracy forwarding with nearly 100 % while the detection rate reaches to 100 % for two cases of IP Spoofing. Besides, the forwarding services time is 0.36 μs and 9.36 μs for the minimum and maximum packet respectively. That leads to the performance of total system reaching 9.859 Gbps in half-duplex and 19.718 Gbps in full-duplex mode. The main contributions of our paper can be summarized as follow.

- We propose a Secured-OFS with integrated security functions. To the best of our knowledge, this is the first OpenFlow switch that can not only route network packets according to the OpenFlow protocol but also defend against network attacks.
- We present our first prototype Secured-OFS using the NetFPGA-10G board which integrates two different DDoS defense mechanisms including Hop-Count Filtering and Port Ingress/Egress Filtering. The first prototype can work at up to 108.711 MHz and achieves a 99.1 % detection rate.

The rest of the paper is organized as follow. Section 2 presents the proposed architecture of Secured-OFS. Section 3 introduces our FPGA-based first prototype version of the proposed Secured-OFS. We analyze our experiments in Sect. 4. Finally, Sect. 5 concludes the paper and introduces future work.

2 Secured-OFS Architecture

In this section, we present our proposed Secured-OFS architecture. Our Secured-OFS can not only operate as an OpenFlow protocol-based switch but also a security device to defense against network attacks.

Figure 1 illustrates our proposed Secured-OFS architecture. The proposed Secured-OFS consists of three different components named *Ingress*, *Egress*, and *Engine*. The Ingress component is responsible for receiving incoming packets from input ports and forwarding to the Engine component for processing. Finally, those packets are routed to corresponding output ports of the Egress component.

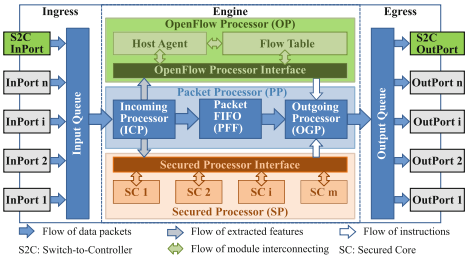


Fig. 1. Secured-OFS architecture

2.1 Ingress Component

The Ingress component includes one input queue, multiple data input ports (*InPort i*), and one control input port (*S2C-InPort*). All incoming packets from these input ports are collected and stored into buffers. Input Queue sequentially selects a packet from the buffer and forwards to the *Engine* component for processing. Input Queue can be configured on the fly so that packets from buffers are selected based on a specific strategy such as Round Robin or input port priorities. Configuration data is sent to Input Queue through the S2C-InPort.

S2C-InPort is the means of communication between a controller and the Secured-OFS. In other words, the corresponding controller sends configuration data through this port to handle the Secured-OFS according to the OpenFlow protocol. Compared to data input ports, this S2C-InPort has a higher priority, i.e., Input Queue selects packets coming from this port to send to the Engine component whenever there is any existence packet in the buffer of S2C-InPort regardless strategies used to select packets at Input Queue.

2.2 Egress Component

In contrast to the Ingress component, Egress consists of an output queue, several data output ports (*OutPort i*), and one control output port (*S2C-OutPort*). A packet after being processed by the Engine component is forwarded to the Output Queue. Regarding to routing information of the packet, Output Queue

sends it to a corresponding data output port. However, following the OpenFlow protocol, there are some cases in which a packet cannot be routed to any data output port due to the lack of information. In those cases, the packet is forwarded to the controller associating with the Secured-OFS through the S2C-OutPort so that the controller can update the Secured-OFS.

2.3 Engine Component

In our proposed architecture, the Engine component plays the most important role. It has much functionality than the works in [11–13]. The component processes an incoming packet that comes from the Ingress component following both OpenFlow protocol and the implementation of network security mechanisms. After processing a packet, the Engine component sends it to the Egress component so that the packet is forwarded to specific destination or the corresponding controller. This component consists of three different processors. Those are OpenFlow Processor (OP), Packet Processor (PP), and Secured Processor (SP).

OpenFlow Processor. The OP consists of a *Host Agent* module, a *Flow Table*, and an interface to communicate with PP. A packet can come to the Secured-OFS through two different port types, the data ports and the control port (S2C-InPort). A packet coming to the Secured-OFS through the control port is a packet generated by the associated controller. This packet contains an instruction that the controller uses to handle the Secured-OFS such as updating Flow Table or modifying a packet header. The Host Agent module is responsible for receiving control packets, executing instructions, and sending feedback to the controller if required when a control packet comes to the Secured-OFS. When a data packet is arriving at the Secured-OFS, it should be scanned by SP to defend against network attacks and processed by this OP following the OpenFlow protocol. The Host Agent module analyzes the data packet and retrieves the Flow Table to extract the corresponding actions associated with the packet. If an action for the packet is found, it will be sent to the PP. Otherwise, OP requests PP to send the packet to the associated controller. To support the OpenFlow protocol, the Flow Table module is needed to store OpenFlow-based actions [14] such as dropping, updating the header fields, or forwarding a packet to a destination output port. Figure 2 illustrates a segment of Flow Table.

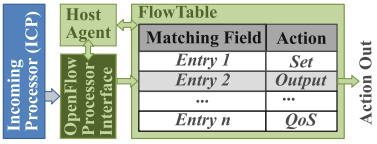


Fig. 2. The Flow Table architecture

Packet Processor. PP consists of three different modules the *Incoming Packet Processing*, the *Packet FIFO*, and the *Outgoing Packet Processing*. It is used to decode an incoming packet into different fields such as header field, and payload field at first. Depending on input port types, PP processes incoming packets in two different scenarios. In the first scenario, a coming packet through the control port is forwarded to the *Host Agent* module of OP without any processing by Incoming Packet Processing module.

In the second scenario, a packet coming through a data input port is analyzed by the Incoming Packet Processing module first. After that both OP and SP are activated simultaneously to process the packet. The header field of the packet is forwarded to the Host Agent module so that corresponding actions of this packet can be found. Depending on which network security mechanisms are used in SP, different fields of the packet are forwarded to SP for scanning. SP guarantees that the packet is a legitimate network packet. The whole packet is stored in the Packet FIFO to wait for decisions from both OP and SP. The Outgoing Packet Processing module starts processing the waiting packet in Packet FIFO immediately after receiving the final decisions from both SP and OP. If SP recognizes that the packet does not belong to a network attack, the taken actions from Flow Table are applied to the packet. Otherwise, the packet is destroyed immediately. In the case of OpenFlow-based actions for a particular packet cannot be found in Flow Table, i.e., the OP requests to send the packet to the associated controller, the Outgoing Packet Processing module forwards this packet to the S2C-OutPort.

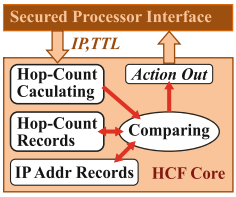
Secured Processor. The SP is responsible for guaranteeing that incoming packets do not belong to network attacks. SP includes a number of *Secured Cores* (SCs) and an interface for communication between the cores and PP. Each Secured Core implements a particular network defense mechanism to defend against a specific type of network attacks such as Anti-DDoS or Anti-Virus. Depending on characteristics of a network where the Secured-OFS is deployed, different security functions are chosen to implement in Secured Cores. Because each Secured Core performs one specific security function, a packet is considered as an illegal packet if there is at least one core alerts the packet is illegal. In this case, a drop signal is issued to PP. Otherwise, when all Secured Cores vote that the packet is legal, a pass signal is sent to the PP.

3 FPGA-Based Secured-OFS Prototype

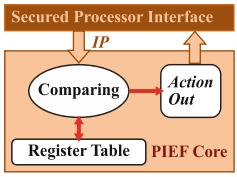
The previous section shows our proposed Secured-OFS architecture. The architecture can be developed by using many different technologies such as FPGA or ASIC. The proposed architecture also can be deployed using multicore systems where each processor in the Engine component can be implemented by a computing core. In this section, we present our first FPGA-based Secured-OFS prototype using the proposed architecture in the previous section.

In this first prototype version, we decide to build two well-known DDoS countering mechanisms, the Hop-Count Filtering (HCF) and the Port Ingress-Egress Filtering (PIEF), in SP because DDoS attacks have become one of the primary cyber-security threats [15,16]. These kinds of attack are attempts to make a computer resource (i.e. website, e-mail, VoIP, or a whole network) unavailable to its intended users. In the final quarter of 2015, the number of DDoS attacks has hit a new record [17].

The Hop-Count Filtering core comprises four main modules called *Hop-Count Calculating*, *Hop-Count Records*, *IP Add Records*, and *Comparing*. Figure 3a presents the architecture of the HCF secured core in this work. When the core receives a source IP address and its final Time-to-Live (TTL) value from PP through Secured Processor Interface, Hop-Count Calculating computes the Hop-Count value for the packet. Then the Comparing module look-ups IP address and stored Hop-Count value of the packet using both the Hop-Count Records and IP Addr Record module. If the calculated Hop-Count value is matched to the stored Hop-Count value, the packet is considered as the legitimate packet. In this case, a *bypass* signal is returned to PP. In the case of the calculated Hop-Count value is different from the stored Hop-Count value, the packet is classified to a DDoS attack. A *drop* signal is issued to alert PP. If this is the first time of a packet coming from this source IP address that has come to the switch, both IP address and TTL value of the packet are stored into Hop-Count Records and the IP Addr Record module. However, this is the main drawback of the Hop-Count defense mechanisms because IP address and TTL value of a packet can be modified while it is traveling in over networks. Therefore, in this work, we consider integrating multiple DDoS countermeasure techniques into a hardware system to improve both protection efficiency and system performance.



(a) HCF Core Architecture



(b) PIEF Core Architecture

Fig. 3. The Secured Cores in our first prototype

Figure 3b depicts the architecture for the PIEF core in this paper. As shown in the figure, the PIEF core contains a Register Table module and a Comparing module. Register Table stores special IP address blocks that are not allowed to appear in networks [18]. The Comparing module compares IP addresses of incoming packets with stored IP addresses in Register Table. When the source IP address of the packet is sent to PIEF, PIEF searches the address in Register Table. If a *miss* signal is returned (i.e., no record was found in Register Table),

the packet is legitimate. Otherwise, the packet is illegitimate (i.e., a *hit* signal is returned). Based on this hit or miss signal, the PIEF core send *bypass* or *drop* signal to PP.

In this first FPGA-based prototype Secured-OFS, the proposed architecture with two DDoS countermeasure mechanisms is built in a single FPGA chip. Therefore, all the inter-components, as well as intra-component interconnects are implemented as on-chip point-to-point and bus-based interconnects. The point-to-point interconnect is very suitable for FPGA implementation because FPGA includes many wires for interconnect. Moreover, the most advantage of the point-to-point interconnect is high-performance because there is no any competition for communication through the point-to-point interconnect. The bus-based interconnect is low latency and area-efficiency. However, bus-based interconnect suffers from low overall communication performance.

We use embedded on-chip memory (Block RAM - BRAM) to build *Flow Table*, *Register Table* in the PIEF secure core, and both *IP Addr Records* and *Hop-Count Records* in the HCF secure core. BRAM allows memory accessing within exactly one cycle so that it helps improve the system performance because of frequent Flow Table accessing. However, BRAM still comprises a disadvantage that is the limitation of usable resources and interconnects. To overcome this disadvantage, external memory chip can be addressed for the prototype of next version.

4 Experiments

In this section, we present our experiments with the first FPGA-based prototype Secured-OFS. We also analyze the hardware resources usage as well as system performance of the proposed Secured-OFS.

4.1 Experimental Setup

In order to validate and estimate the system performance of the first prototype version of our Secured-OFS, we employ two NetFPGA-10G [10] boards to build a testing system. Figure 4 illustrates our testing model. The first NetFPGA-10G board is configured as our FPGA-based Secured-OFS with four data ports using SFP + (Small Form-Factor Pluggable) interface. The PCI Express connector of the board is used as control ports (S2CInPort and S2COutPort) to communicate with OpenFlow Controller software running on the host. The second NetFPGA-10G board is configured as a *Test Agent* in which we port the framework of Open Source Network Test (OSNT) [19]. OSNT including a Generator and a Monitor can be used to generate packets under some parameters and monitor incoming network packets. The Generator is used to generate not only legitimate packets but also attacking packets at the line-rate ($\approx$ 10Gbps/port) to send testing data to the input ports of our Secured-OFS system. All legitimate packets are switched to one of three output ports of Secured-OFS from *SFP+Port0-Out* to *SFP+Port2-Out* depending on the handling of the controller. Instead of dropping

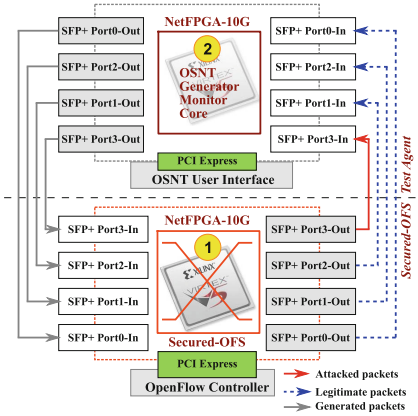


Fig. 4. The testing model of Secured-OFS

attacking packets as presented above, we configure the Secured-OFS to forward those packets to *SFP+Port3-Out* for doing statistic.

In our experiments, we conduct several test cases with different packets in term of size, header, and payload data. We measure and calculate the interval to process the *first-packet* coming from a specific source and the switching time of *non-first-packet* to validate the OpenFlow operation of the proposed Secured-OFS. When the first packet comes from a specific source, switching information of this packet does not exist in Flow Table. Following the OpenFlow protocol, the switch needs to forward it to the associated controller. We also collect data from the report of OSNT Monitor in which all packets are classified to check the capacity of DDoS protection of the secured cores such as detection rate, false positive rate, and false negative rate. Finally, we use several packet sizes to evaluate overall throughput of the proposed Secured-OFS.

4.2 Experiment Results

Table 1 summarizes hardware resources usage and synthesis frequency of the prototype Secured-OFS. Due to the fact that we use BRAM to implement Flow Table in OpenFlow Processor, IP Addr Records and Hop-Count Records in the HCF core, and Register Table in the PIEF core, the Secured-OFS consumes much more BRAM resource than other types. The first prototype version of our Secured-OFS can work at up to 108.711 MHz.

In performance evaluation experiment, generated packets belong to one of six types 64 bytes, 128 bytes, 256 bytes, 512 bytes, 1024 bytes, and 1500 bytes. The range from 64 bytes to 1500 bytes is valid for an Ethernet packet. We use these packet to test the OpenFlow protocol function of the switch in both half-duplex and full-duplex mode. The results of this performance evaluation experiment is shown in Fig. 5. In this chart, the horizontal axis indicates the *Packet size* in *bytes* while the vertical axis shows *Throughput* in *in Gbps* responding to each

Table 1. The hardware resource utilization of the system

Resources	Amount	Percentage
Look Up Tables (LUTs)	54770	36 %
Registers (FFs)	57424	38 %
Block RAM	204	62 %

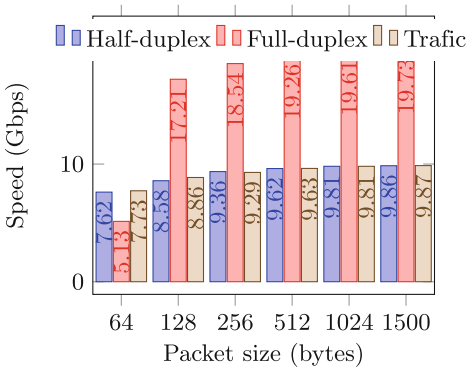


Fig. 5. Performance testing of Secured-OFS

packet types. In each packet type, the first column shows system throughput in the half-duplex mode. The second column depicts throughput in full-duplex mode. Finally, the last column presents OSNT packet generator speed. According to these experimental results, our first prototype system can achieve throughput by up to 9.859 Gbps in half-duplex mode and up to 19.729 Gbps in full-duplex mode.

In the switching time experiments, each flits in a packet (a packet is divided into a number of fixed-size flits) requires 15 cycles to be processed. Therefore, total forwarding time for a packet depends on the packet size. Table 2 presents processing time for each packet type. A 64-byte packet consumes 0.36 μs while a 1500-byte packet requires 9.1 μs to be processed. The proposed Secured-OFS is also validated in the case of first packet coming from a specific source arriving the switch. The experiments show that the functionality of our Secured-OFS is fully satisfied. Table 3 shows the detection rates, false positive rates, and false negative rates according to the packet sizes. The detection rates, false negative rates, and false positive rates are almost stable when the sizes of packets are changed. According to the table, the system can recognize all attacking packets although a 2.9 % false negative rate (legitimate packets are classified as attacking packets) occurs during the test.

Table 2. Timing testing of Secured-OFS

Packet size (bytes)	Forwarding time (μs)	First packet delay Time (μs)
64	0.36	0.08
128	0.7	0.14
256	1.38	0.26
512	2.76	0.51
1024	5.36	0.98
1500	9.36	1.78

Table 3. Detection rate of Secured-OFS *(in %)*

Type	<64	64–128	128–256	256–512	>512
Detection rate	97.1	97.6	98.4	98.6	99.1
False negative	0.0	0.0	0.0	0.0	0.0
False positive	2.9	2.4	1.6	1.4	0.9

5 Conclusion

In this paper, we propose a novel architecture for an OpenFlow switch which can defend against vulnerabilities from the network. This research contributes the novel framework which has high expansibility due to the Secured-OFS system based-on the combination with many secured functions. This architecture contains a processor able to merge many different secured core results into a final decision to against network intrusions. In the implementation, we integrate the HCF core and the PIEF core into Secured-OFS to achieve the high detection rate with DDoS attack. We have the plan to extend our Secured-OFS by integrating a dynamic reconfiguration module to this system. That helps the system become more flexible and efficient because the system could change secured cores to adapt to network attack types.

Acknowledgement. This research is funded by Ho Chi Minh City University of Technology (HCMUT) under grant number TSDH-2015-KHMT-52.

References

1. Goransson, P., Black, C.: Software Defined Networks: A Comprehensive Approach. Morgan Kaufmann, San Francisco (2014)
2. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, P., Peterson, L., Rexford, J., Shenker, S., Turner, J.: OpenFlow: enabling innovation in campus networks, vol. 38, pp. 69–74. ACM, New York (2008)
3. Gelberger, A., Yemini, N., Giladi, R.: Performance analysis of software-defined networking (SDN). In: 2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems. IEEE Press, San Francisco (2013)

4. Kreutz, D., Ramos, F., Verissimo, P.: Towards secure and dependable software-defined networks. In: Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, pp. 55–60. ACM, New York (2013)
5. Tootoonchian, A., Ganjali, Y.: HyperFlow: a distributed control plane for OpenFlow. In: Proceedings of the 2010 Internet Network Management Conference on Research on Enterprise Networking. USENIX Association, Berkeley (2010)
6. Braga, R., Mota, E., Passito, A.: Lightweight DDoS flooding attack detection using NOX/OpenFlow. In: 2010 IEEE 35th Conference on Local Computer Networks (LCN), pp. 408–415. IEEE, Denver (2010)
7. Shin, S., Song, Y., Lee, T., Lee, S., Chung, J., Porras, P., Yegneswaran, V., Noh, J., Kang, B.B.: Rosemary: a robust, secure, and high-performance network operating system. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, pp. 78–89. ACM, New York (2014)
8. Hu, Y., Wang, W., Gong, X., Que, X., Cheng, S.: Balanceflow: controller load balancing for openflow networks. In: 2012 IEEE 2nd International Conference on Cloud Computing and Intelligent Systems (CCIS), vol. 2, pp. 780–785. IEEE, New York (2012)
9. Chen, X., Zheo, B., Ma, S., Chen, C., Hu, D., Zhou, W., Zhu, Z.: Leveraging master-slave OpenFlow controller arrangement to improve control plane resiliency in SD-EONs. In: Optics Express, vol. 23, no. 6, pp. 7550–7558. Optical Society of America (2015)
10. NetFPGA 10G. http://netfpga.org/site/#/systems/3netfpga-10g/details/
11. Naous, J., Erickson, D., Covington, G.A., Yang, S., Appenzeller, G., McKeown, N.: Implementing an OpenFlow switch on the NetFPGA platform. In: Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems, pp. 1–9. ACM, New York (2008)
12. OpenFlow Implementation on NetFPGA-10G. https://docs.google.com/document/d/1ZwHXQZocKwQls6Ted8VZO8h9MjBtu9WxV2fAY44eOgE/edit
13. Naous, J., Erickson, D., Covington, G.A., Yang, S., Appenzeller, G., McKeown, N.: From 1G to 10G: code reuse in action. In: Proceedings of High performance and programmable networking, pp. 31–38. ACM, New York (2013)
14. Suh, M., Park, S.H., Lee, B., Yang, S.: Building firewall over the software-defined network controller. In: 16th International Conference on Advanced Communication Technology, pp. 744–748. IEEE, Pyeongchang (2014)
15. Zargar, S.T., Joshi, J., Tipper, D.: A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks. In: IEEE Communications Surveys & Tutorials, vol. 15, no. 4, pp. 2046–2069. IEEE (2013)
16. Yan, Q., Yu, F.R., Gong, Q., Li, J.: Software-defined networking (SDN) and distributed denial of service (DDoS) attacks in cloud computing environments: a survey, some research issues, and challenges. In: IEEE Communications Surveys & Tutorials, vol. 18, pp. 602–622. IEEE (2016)
17. State of the Internet Report (2016). https://www.stateoftheinternet.com/resources-cloud-security-2015-Q4-web-security-report.html
18. Cotton, M., Vegoda, L.: Special Use IPv4 Addresses. Technical report, RFC-57 (2010)
19. The open source network tester. http://osnt.org/