DoS attack mitigation in SDN networks using a deeply programmable packet-switching node based on a hybrid FPGA/CPU data plane architecture

Enio Kaljic
Department of Telecommunications
Faculty of Electrical Engineering
University of Sarajevo
Sarajevo, Bosnia and Herzegovina
enio.kaljic@etf.unsa.ba

Almir Maric
Department of Telecommunications
Faculty of Electrical Engineering
University of Sarajevo
Sarajevo, Bosnia and Herzegovina
almir.maric@etf.unsa.ba

Pamela Njemcevic
Department of Telecommunications
Faculty of Electrical Engineering
University of Sarajevo
Sarajevo, Bosnia and Herzegovina
pamela.njemcevic@etf.unsa.ba

***Abstract*—The application of the concept of software-defined networks (SDN) has, on the one hand, led to the simplification and reduction of switches price, and on the other hand, has created a significant number of problems related to the security of the SDN network. In several studies was noted that these problems are related to the lack of flexibility and programmability of the data plane, which is likely first to suffer potential denial-of-service (DoS) attacks. One possible way to overcome this problem is to increase the flexibility of the data plane by increasing the depth of programmability of the packet-switching nodes below the level of flow table management. Therefore, this paper investigates the opportunity of using the architecture of deeply programmable packet-switching nodes (DPPSN) in the implementation of a firewall. Then, an architectural model of the firewall based on a hybrid FPGA/CPU data plane architecture has been proposed and implemented. Realized firewall supports three models of DoS attacks mitigation: DoS traffic filtering on the output interface, DoS traffic filtering on the input interface, and DoS attack redirection to the honeypot. Experimental evaluation of the implemented firewall has shown that DoS traffic filtering at the input interface is the best strategy for DoS attack mitigation, which justified the application of the concept of deep network programmability.**

***Index Terms*—DoS, DDoS, firewall, SDN, software-defined networking, data plane, DPN, FPGA, 5G**

I. Introduction

Software-defined networks (SDN) were created as an answer to the problems of management and maintenance of communications networks. Unlike traditional networks where control (control plane) and packet forwarding (data plane) processes take place within each network element, SDN separates these processes and executes them in separate network entities. All network intelligence executes in the control plane, in the so-called controllers, and the packet-switching nodes (switches, routers, etc.) carry out simple traffic forwarding according to the control plane instructions. Standardization of the SDN architecture began with the specification of ForCES [1] and its practical implementation with the introduction of OpenFlow [2].

The security features of OpenFlow based networks are usually implemented in such way that each time when a new traffic flow arrives at the switch, the first packet of that flow is delivered to the controller which performs security checks and determines how to proceed with that flow. However, it is often not possible to make the right decision based on only one packet, the first packet, which makes it difficult to integrate security procedures into the standard OpenFlow model. Because of this, complex security mechanisms are most often implemented as middlebox functionality [3], and the OpenFlow based network serves solely to route traffic to the dedicated middlebox. The use of middleboxes, on the one hand, favors OpenFlow based networks because it does not require the security treatment of traffic on controllers, and on the other hand, opens the possibility of attacks on network infrastructure (switches and controllers) located between attackers and middleboxes [4]. To overcome these problems, in a large number of studies, reviewed in [5]–[7], have been proposed architectural changes to the OpenFlow model that enable the implementation of security functionalities directly in the data plane.

From the above examples, it can be concluded that the application of OpenFlow based SDN architecture has, on the one hand, led to simplification and reduction of cost of packet-switching nodes and, on the other hand, created a substantial number of problems in terms of data plane flexibility and programmability, as is elaborated in [7]. When it comes to the security of SDN based networks, there are some limitations of ForCES and OpenFlow data plane architectures, which reflects in the inability of state-aware or context-aware packet processing, lack of support for packet classification on higher-layer protocols (e.g. SIP, HTTP) and the inability to use hybrid architectures for offloading of data plane processes. One possible way to overcome identified problems is to increase the data plane flexibility by applying the concept of deep network programmability, which is proposed in [8] and experimentally demonstrated in [9]. In these papers, deep network programmability means the introduction of support for full programmability of all data plane processes below the flow table lookup, including processes outside the packet processing

pipeline (e.g., queue scheduler, physical interface controller). The deeply programmable packet-switching node (DPPSN), presented in [8], [9], is based on a hybrid FPGA/CPU data plane architecture. Based on the proposed architecture, a reference 10Gbps Ethernet switch has been implemented.

Considering the above, the goal of this paper is to reduce the negative impacts of denial-of-service (DoS) attacks in SDN networks by applying the concept of deep network programmability. The set goal has been achieved by completing the following tasks, which also represent the contributions of this paper:

- feasibility of using a reference switch architecture based on the DPPSN in the implementation of a firewall has been investigated,
- an architectural model of the firewall, which supports three models of DoS attacks mitigation – (1) DoS traffic filtering on the output interface, (2) DoS traffic filtering on the input interface, and (3) DoS attack redirection to the honeypot, has been proposed,
- proposed architectural model of the firewall has been implemented using hybrid FPGA/CPU data plane architecture, and experimentally verified using software and hardware tools for generating and analyzing network traffic,

The rest of this paper is organized as follows. Section II provides an overview of related research which attempted to solve the problem of DoS attacks mitigation in SDN networks by relocating firewall logic from control to data plane. Then, in Section III, a DPPSN-based firewall architecture has been proposed, and practical implementation based on hybrid FPGA/CPU data plane architecture presented. Section IV presents experimental verification of the practical firewall implementation, proceeding with a discussion of obtained results. Finally, Section V provides the conclusion and suggestion for future research.

II. Related work

Although pure SDN advocates the clear separation of data plane and control plane processes, advantages of retaining certain control plane processes in the data plane in realizing network security functionalities have been investigated in many papers. Thus, in [10], an extension of the OpenFlow data plane, named Avant-Guard, has been introduced to reduce the number of controller-switch interactions during a DoS attack. To achieve this, actuating triggers, which allow detection of attacks and the rapid installation of new traffic forwarding rules, have been introduced into the data plane above existing statistical counters. Based on the proposed solution, a firewall for protection against TCP SYN flood attacks has been implemented using a software-based OpenFlow reference switch. Driven by the same motives, in [11] has been proposed a security framework, called StateFit, which enables state-based traffic filtering in the data plane. Based on the StateFit framework, a prototype firewall which protects against ICMP flood attacks has been modeled using the P4 language and implemented using a BMv2-based software switch. Similar to the above solution, a new SDN's data plane architecture which allows the relocation of stateful firewall logic from the control plane to the data plane has been proposed in [12].

In addition to software-based firewall implementations, a considerable number of papers advocate the use of reconfigurable hardware (such as FPGA) to achieve better performance than software implementations. Some FPGA-based implementations of firewall elements (e.g., attack detection, packet filtering), which may be useful in implementing a complete firewall, are presented in [13]–[16]. NetFPGA [17] and NetFPGA-SUME [18] are the most commonly used platforms. In [19] has been introduced a hardware-based system for defending against HTTP GET flood attacks. The prototype of the system has been implemented by data plane extension of the NetFPGA based OpenFlow switch [20] with elements for HTTP GET flood attacks detection and mitigation. Similar to the previous solution, in [21] has been introduced a new model of the OpenFlow switch pipeline which supports defense against TCP SYN flood attacks. The prototype has been implemented using the NetFPGA-10G platform. In [22] has been proposed a generic solution to the SDN based data plane architecture, called DPX, which natively supports security services as a set of abstract security actions translated into OpenFlow rules. The DPX firewall prototype has been implemented using the NetFPGA-SUME platform. Especially interesting is the use of FPGA technology in accelerating the execution of security functionalities in SDN-based 5G mobile networks, which are subject to strict delay and bandwidth requirements.Thus, in [23] has been introduced the implementation of a 5G firewall, which enables detection, differentiation and selective blocking of DoS traffic in the edge-to-core segment of the 5G network infrastructure. The prototype of the proposed 5G firewall has been implemented using the P4 language [24] and the NetFPGA-SUME platform.

Given that pure hardware-based firewall implementations suffer from the lack of flexibility to support detection and mitigation of new types of attacks, offloading of pre-detection and coarse attack detection functionalities from the control plane to the hybrid FPGA/CPU data plane has been proposed in [25], [26]. The software part of the data plane running on the CPU is in charge of detecting attacks and generating new traffic forwarding rules, and the hardware part of the data plane implemented on the FPGA is in charge for packet forwarding. The proposed solution is based on the assumption that the CPU inside the OpenFlow switch is underutilized and capable of taking on those additional tasks, which turns out to be true for 1Gbps Ethernet.

When summarizing related work, it can be concluded that the use of hybrid FPGA/CPU architectures in implementation of packet-switching nodes can contribute to the data plane programmability. However, the advantages and disadvantages of using hybrid FPGA/CPU technologies in the implementation of SDN based firewalls for communication speeds above 10Gbps are still poorly known, further confirming the motivation of this paper.

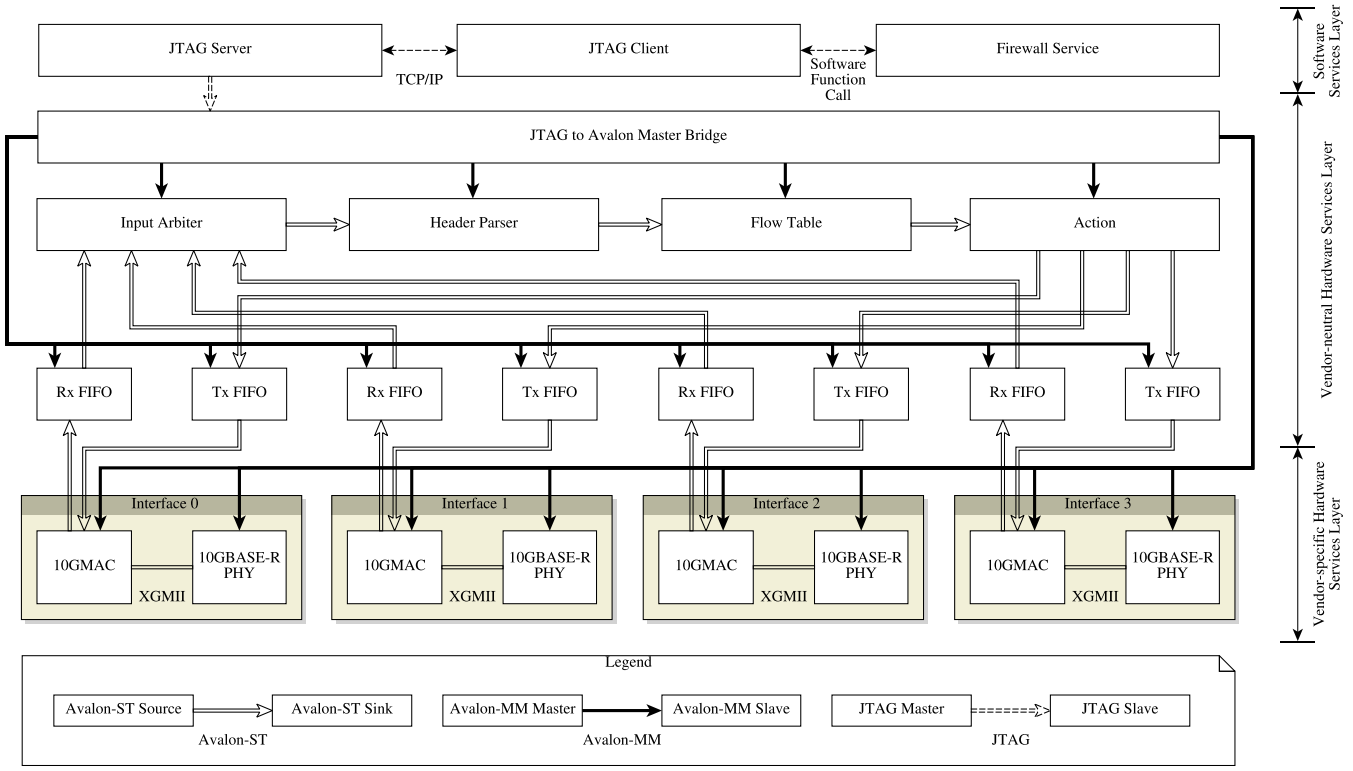


Figure 1. Firewall architecture based on a deeply programmable packet-switching node

III. Proposed solution

An analysis of the DPPSN based reference switch architecture shown that functionalities of the SDN based firewall can be implemented by retention and upgrade of existing switching processes. Figure 1 shows the proposed hybrid FPGA/CPU architecture of the firewall consisting of three layers:

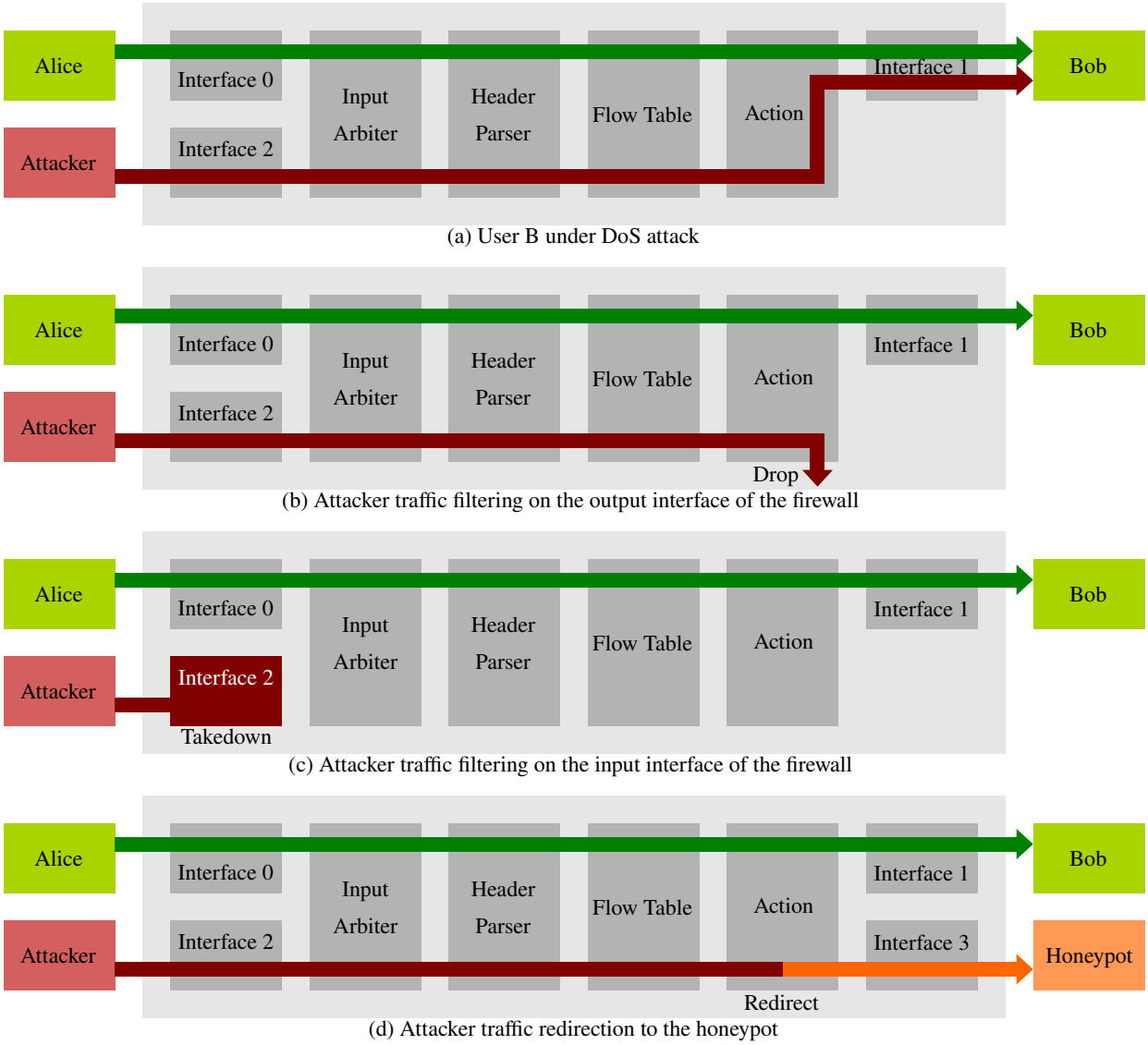
- software services layer – which contains services implemented as software applications,
- vendor-neutral hardware services layer – which contains services implemented as instances of user-defined modules on an FPGA chip,
- vendor-specific hardware services layer – which contains hardware modules specific to the FPGA chip manufacturer.

The following changes have been made to the hardware services layer compared to the reference switch architecture. The processes of input queues scheduling and packet header parsing are entirely taken from the reference switch. The flow table has been implemented by extending the match-action table from the reference switch to support the functionality of TCP, UDP, and ICMP traffic classification with the programmable header parser. The output demultiplexer has been extended with a packet drop action. The connection between the layers of hardware and software services has been realized using the JTAG interface. In the software services layer, in addition to the JTAG to TCP/IP communication adapter, the functionality of the firewall service has been implemented. Details regarding the implementation of the proposed architecture are given bellow.

The hardware services layer of the proposed firewall has been implemented using the Altera DE5-Net FPGA development board, and the JTAG interface to the CPU has been implemented using an on-board USB Blaster II programmer. Support for 10G Ethernet network technology has been implemented in the vendor-specific hardware services layer using embedded 10Gbps serial transceivers (10GBASE-SR PHY) and an IP core for 10G MAC. The hardware services modules are interconnected using the Avalon-ST interface for packet transfer and the Avalon-MM interface for sporadic transfer of configuration data from and to the software services layer. The firewall service has been implemented as a Java application, which communicates with the hardware services layer using a JTAG client.

The principle of operation of the proposed firewall is as follows. For new traffic flows, according to the controller's instructions, new forwarding rules are installed in the flow table. Since the firewall does not involve the controller in its operation afterward, the controller is omitted from further analysis. The firewall service periodically monitors the match counters in the flow table, thus recognizing the activity of each traffic flow. In this paper, the functionality of UDP flood attack detection has been implemented. The attacked site behaves as follows:

1) checks if there is a destination application,



(a) User B under DoS attack

(b) Attacker traffic filtering on the output interface of the firewall

(c) Attacker traffic filtering on the input interface of the firewall

(d) Attacker traffic redirection to the honeypot

Figure 2. Experiment scenarios

2) determines that there is no destination application,
3) responds with the ICMP Destination Unreachable packet.

Therefore, the attack detection functionality has been realized by monitoring the change of the counter value for the identified UDP traffic flow and the returning ICMP traffic flow. When it is observed that an increment in the ICMP flow counter is proportional to the increment in the UDP flow counter, an attack is detected. After the attack has been detected, and according to the start-up configuration, the firewall can operate in one of three ways:

1) to install a new rule in the flow table to drop attacker traffic just before the output interface – DROP strategy,
2) to reconfigure the input interface to filter attacker traffic – TAKEDOWN strategy,
3) to install a new rule in the flow table to redirect attacker traffic to the honeypot – REDIRECT strategy.

Strategies labeled with DROP and REDIRECT can be easily implemented in SDN based switches because both are based on the flow table content modification, while the TAKEDOWN strategy requires the application of the deep network programmability concept. That justifies the use of DPPSN to implement the firewall.

Although, in this paper, only the detection and mitigation of UDP flood attacks have been implemented to demonstrate the operation of the firewall, support for other types of DoS attacks can be implemented relatively easily by modifying the software service only. For example, ICMP flood attack detection can be implemented by monitoring the change of the ICMP Echo Request packet counter.

IV. Experiment

This section describes the experimental evaluation of firewalls in detecting and mitigating UDP flood attacks using

Table I
FORWARDING THROUGHPUT OF FIREWALL

Case	Throughput			
	Maximum (Gbps)	*Minimum (Gbps)*	*Average (Gbps)*	*Relative (%)*
Normal operation	9.57	9.31	9.39	100.00
Under attack	9.49	0.00	6.82	72.60
DROP	9.42	0.00	7.01	74.67
TAKEDOWN	9.66	8.28	9.39	99.95
REDIRECT	9.56	0.00	7.02	74.76

the three strategies outlined above. Four 10GbE firewall interfaces are connected to four workstations which mimic the functionality of two users, referred below as Alice and Bob, one attacker and one honeypot. Figure 2 shows four scenarios of conducted experiments.

In the first scenario, Alice and Bob exchange data using the TCP protocol, making very efficient use of the available network bandwidth. At time t_1 = 10 seconds, the attacker starts an UDP flood attack against Bob, and at time t_2 = 20 seconds, the attack stops. The FlueNT10G [27], a programmable network tester implemented with the NetFPGA-SUME development platform, was used to generate test traffic and accurately measure latency and throughput. The *hping* tool was used to perform the DoS attack. In the first scenario, attack detection and mitigation functionalities are disabled.

The second, third, and fourth scenarios demonstrate three strategies for mitigating attacks: DROP, TAKEDOWN, and REDIRECT, respectively. As described earlier, the DROP strategy involves filtering attacker traffic on the firewall output interface, which is implemented by installing a rule in a flow table that rejects attacker traffic. On the other hand, the TAKEDOWN strategy involves filtering the attacker traffic at the very entrance to the firewall, which has the advantage of reducing the firewall pipeline load with unwanted traffic. In the case of REDIRECT strategy, attacker traffic is diverted to the honeypot instead of discarded. The advantage of DROP and REDIRECT strategies over TAKEDOWN lies in the ease of implementation in SDN based packet-switching nodes since it does not require any interventions on the firewall input interfaces.

Table I shows the firewall throughput for normal operation of the network, and for each of the four described scenarios. For these four scenarios, a relative throughput, defined as the ratio of achieved throughput to the throughput of the firewall without DoS attack, is also shown.

Figure 3 shows the change of firewall throughput in a time window of 30 seconds, where the attack begins in t_1 = 10 seconds and ends in t_2 = 20 seconds. Since the throughput for the TAKEDOWN strategy is not clearly visible in the preceding graph, it is separately shown in Figure 4.

In addition to measuring firewall throughput, traffic forwarding latency, which does not include any additional processing delays on the client and server side, has been measured. The obtained results are shown in Figure 5.

From the obtained firewall throughput measurements, it

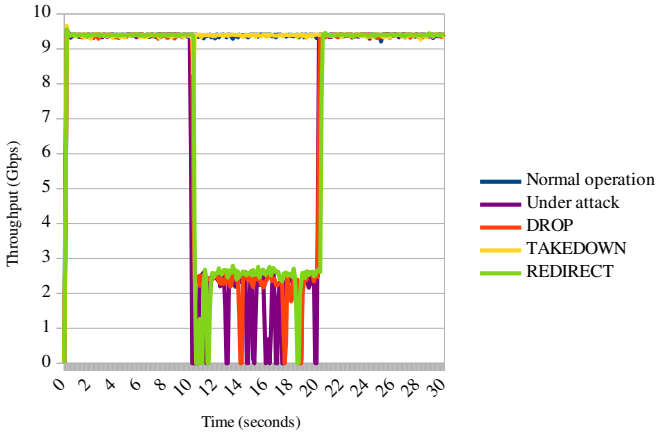


Figure 3. Firewall forwarding throughput

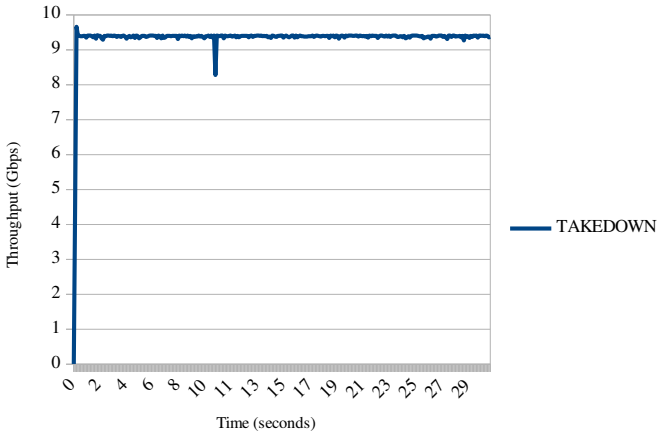


Figure 4. Firewall forwarding throughput – TAKEDOWN strategy

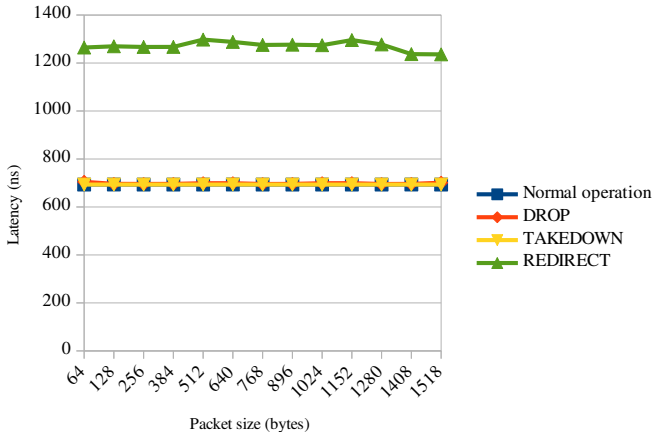


Figure 5. Firewall forwarding latency

can be seen that the gains of the DROP and REDIRECT strategies, although the simplest to implement, are very small compared to the network operation without protection. In terms of forwarding latency, the REDIRECT strategy proved to be the worst. It can be concluded that its use is justified only on the assumption that the subjective feeling of the

attacker about the success of the attack will deter him from continuing the attack, or in case of need for forensic processing of the attacker. TAKEDOWN has proven to be by far the best strategy, which justified the application of the concept of deep network programmability.

V. Conclusion

This paper proposes an SDN firewall architecture, based on a deeply programmable packet-switching node, which supports DoS attack mitigation using three strategies: (1) DROP – DoS traffic filtering on the output interface, (2) TAKEDOWN – DoS traffic filtering on the input interface, and (3) REDIRECT – DoS traffic redirection to the honeypot. The practical implementation of the firewall has been done using a hybrid FPGA/CPU architecture. An experimental evaluation of the firewall has been conducted using a programmable network tester to generate user traffic and software tools to perform DoS attacks. The obtained results showed that DoS traffic filtering on the firewall input interface is the best strategy, whose implementation is based on the deep network programmability concept. That confirmed that the set goal of the work has been achieved – reducing the negative impacts of DoS attacks in SDN networks by applying the concept of deep network programmability.

Considering that in this paper only detection and mitigation of UDP flood attacks have been implemented to evaluate firewall, for future work it can be interesting to implement support for detection and mitigation of other types of DoS attacks. It may also be interesting to investigate the feasibility of integrating the proposed firewall architecture into the SDN based edge-to-core network infrastructure of 5G mobile networks.

References

[1] H. M. Khosravi and T. A. Anderson, "Requirements for separation of IP control and forwarding," RFC 3654, Dec. 2003, accessed: 2018-10-12. [Online]. Available: https://rfc-editor.org/rfc/rfc3654.txt

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[3] A. Gember, P. Prabhu, Z. Ghadiyali, and A. Akella, "Toward software-defined middlebox networking," in *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, ser. HotNets-XI. New York, NY, USA: ACM, 2012, pp. 7–12.

[4] K. Benzekki, A. El Fergougui, and A. Elbelrhiti Elalaoui, "Software-defined networking (SDN): a survey," *Security and Communication Networks*, vol. 9, no. 18, pp. 5803–5833, 2016, sCN-16-0386.R1.

[5] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, Jan 2015.

[6] T. Dargahi, A. Caponi, M. Ambrosin, G. Bianchi, and M. Conti, "A survey on the security of stateful SDN data planes," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1701–1725, thirdquarter 2017.

[7] E. Kaljic, A. Maric, P. Njemcevic, and M. Hadzialic, "A survey on data plane flexibility and programmability in software-defined networking," *IEEE Access*, vol. 7, pp. 47 804–47 840, 2019.

[8] E. Kaljic, A. Maric, and M. Hadzialic, "A novel qualitative metric based approach to the improvement of data plane flexibility in software-defined networks," in *IEEE EUROCON 2019 - 18th International Conference on Smart Technologies*, Novi Sad, Jul. 2019.

[9] E. Kaljic, A. Maric, and P. Njemcevic, "An implementation of a deeply programmable SDN switch based on a hybrid FPGA/CPU architecture," in *2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH)*, Mar. 2019, pp. 1–6.

[10] S. Shin, V. Yegneswaran, P. Porras, and G. Gu, "AVANT-GUARD: Scalable and vigilant switch flow management in software-defined networks," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS '13. New York, NY, USA: ACM, 2013, pp. 413–424.

[11] R. Hwang, V. Nguyen, and P. Lin, "StateFit: A security framework for SDN programmable data plane model," in *2018 15th International Symposium on Pervasive Systems, Algorithms and Networks (I-SPAN)*, Oct 2018, pp. 168–173.

[12] M. Caprolu, S. Raponi, and R. Di Pietro, "FORTRESS: An Efficient and Distributed Firewall for Stateful Data Plane SDN," *Security and Communication Networks*, vol. 2019, p. 16, 2019.

[13] S. Ghanti and G. M. Naik, "Efficient Data Transfer Rate and Speed of Secured Ethernet Interface System," *International Scholarly Research Notices*, vol. 2016, p. 8, 2016.

[14] A. Fiessler, S. Hager, B. Scheuermann, and A. W. Moore, "HyPaFilter - a versatile hybrid FPGA packet filter," in *2016 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*, March 2016, pp. 25–36.

[15] N. Hoque, H. Kashyap, and D. Bhattacharyya, "Real-time DDoS attack detection using FPGA," *Computer Communications*, vol. 110, pp. 48 – 58, 2017.

[16] B. Nagy, P. Orosz, and P. Varga, "Low-reaction time FPGA-based DDoS detector," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, April 2018, pp. 1–2.

[17] G. Watson, N. McKeown, and M. Casado, "NetFPGA: A tool for network research and education," in *2nd workshop on Architectural Research using FPGA Platforms (WARFP)*, vol. 3, 2006.

[18] N. Zilberman, Y. Audzevich, G. A. Covington, and A. W. Moore, "NetFPGA SUME: Toward 100 Gbps as research commodity," *IEEE Micro*, vol. 34, no. 5, pp. 32–41, Sept 2014.

[19] A. N. Viet, L. P. Van, H. N. Minh, H. D. Xuan, N. P. Ngoc, and T. N. Huu, "Mitigating HTTP GET flooding attacks in SDN using NetFPGA-based OpenFlow switch," in *2017 14th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, June 2017, pp. 660–663.

[20] J. Naous, D. Erickson, G. A. Covington, G. Appenzeller, and N. McKeown, "Implementing an OpenFlow switch on the NetFPGA platform," in *Proceedings of the 4th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, ser. ANCS '08. New York, NY, USA: ACM, 2008, pp. 1–9.

[21] D.-M. Ngo, C. Pham-Quoc, and T. Ngoc Thinh, "An Efficient High-Throughput and Low-Latency SYN Flood Defender for High-Speed Networks," *Security and Communication Networks*, vol. 2018, p. 14, 2018.

[22] T. Park, Y. Kim, V. Yegneswaran, P. Porras, Z. Xu, K. Park, and S. Shin, "DPX: Data-Plane eXtensions for SDN Security Service Instantiation," in *Detection of Intrusions and Malware, and Vulnerability Assessment*, R. Perdisci, C. Maurice, G. Giacinto, and M. Almgren, Eds. Cham: Springer International Publishing, 2019, pp. 415–437.

[23] R. Ricart-Sanchez, P. Malagon, J. M. Alcaraz-Calero, and Q. Wang, "Hardware-Accelerated Firewall for 5G Mobile Networks," in *2018 IEEE 26th International Conference on Network Protocols (ICNP)*, Sep. 2018, pp. 446–447.

[24] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese, and D. Walker, "P4: Programming protocol-independent packet processors," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, Jul. 2014.

[25] X. Yang, B. Han, Z. Sun, and J. Huang, "SDN-Based DDoS Attack Detection with Cross-Plane Collaboration and Lightweight Flow Monitoring," in *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, Dec 2017, pp. 1–6.

[26] B. Han, X. Yang, Z. Sun, J. Huang, and J. Su, "OverWatch: A Cross-Plane DDoS Attack Defense Framework with Collaborative Intelligence in SDN," *Security and Communication Networks*, vol. 2018, p. 15, 2018.

[27] A. Oeldemann, T. Wild, and A. Herkersdorf, "FlueNT10G: A Programmable FPGA-based Network Tester for Multi-10-Gigabit Ethernet," in *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*, Aug 2018, pp. 178–1787.