Accelerated DDoS Attacks Mitigation using Programmable Data Plane

Mário Kuka
CESNET, a.l.e.
Prague, Czech Republic
kuka@cesnet.cz

Kamil Vojanec
Brno University of Technology
Faculty of Information Technology
Centre of Excellence IT4Innovations
Brno, Czech Republic
xvojan00@stud.fit.vutbr.cz

Jan Kučera
CESNET, a.l.e.
Prague, Czech Republic
jan.kucera@cesnet.cz

Pavel Benáček
CESNET, a.l.e.
Prague, Czech Republic
benacek@cesnet.cz

***Abstract*—DDoS attacks are a significant threat to internet service or infrastructure providers. This poster presents an FPGA-accelerated device and DDoS mitigation technique to overcome such attacks. Our work addresses amplification attacks whose goal is to generate enough traffic to saturate the victims links. The main idea of the device is to efficiently filter malicious traffic at high-speeds directly in the backbone infrastructure before it even reaches the victim's network. We implemented our solution for two FPGA platforms using the high-level description in P4, and we report on its performance in terms of throughput and hardware resources.**

***Index Terms*—Denial-of-service Attacks, DDoS Mitigation, Programmable Data Planes, P4, FPGA**

I. Introduction

Distributed Denial-of-Service (DDoS) attacks are one of the most serious threats to all internet service or infrastructure providers. Such an attack aims to take down a service or even the whole network to make it inaccessible to legitimate users. During the attack, the network traffic generally consists of legitimate and malicious packets. The main issue is that the malicious packets consume bandwidth and other resources of the victim so that other incoming packets, especially the legitimate ones, have to be discarded non-deterministically. To mitigate the attack it is necessary to find a way to efficiently distinguish between a legitimate and a malicious packet with minimal disruption to the communication of legitimate traffic. However, due to the overwhelming computational complexity of the mitigation methods, this leads to challenges in meeting the performance requirements of modern high-speed networks. Moreover, the volume and complexity of DDoS attacks continuously grows every year [1], [2] which makes efficient defense more and more difficult and expensive.

This poster aims to present our DDoS mitigation technique. We propose an FPGA-accelerated device which deals with such attacks by filtering the traffic in an ISP backbone infrastructure before it reaches the victim's network. The proposed mitigation algorithm focuses on reflection attacks [3], while the decision to block the traffic is based on the volumetric contribution of source IP addresses to the attack. Originally, we implemented our idea manually using pure VHDL. Later, we also ported this architecture into a high-level description using P4 language [4], [5] based on match-action tables. Finally, we generated a synthesizable code suitable for two FPGA platforms, i.e. Xilinx UltraScale+ and Intel Stratix 10. In this work, we present the details of architecture design and then report its performance in terms of throughput and its requirements in terms of hardware resources.

This research has been supported by the Ministry of Education, Youth and Sports of the Czech Republic from the National Programme of Sustainability (NPU II), project IT4Innovations excellence in science – LQ1602, and by the project Reg. No. CZ.02.1.01/0.0/0.0/16_013/0001797 co-funded by the Ministry of Education, Youth and Sports of the Czech Republic and European Regional Development Fund.

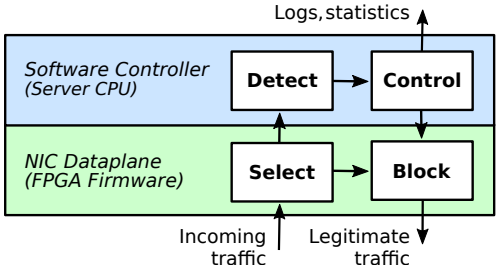

Fig. 1. High-level architecture of the device.

II. System Design

The proposed mitigation device uses a commodity hardware server with a dedicated 100 GbE network interface card connected to the host via PCI Express bus and equipped with an FPGA chip. The high-level architecture of the device is depicted in Fig. 1.

NIC implements fast packet forwarding and filtering data plane managed by a software controller. According to a set of user-defined rules, the FPGA accelerator first selects affected subset of incoming network traffic (a destination IP prefix, DNS traffic, etc.), extracts and collects interesting data from packets (header fields, e.g., IP addresses, TCP/UDP ports, volume statistics) and provides these data in a predefined format to the software part of the system. The controller continuously evaluates these network traffic parameters, and as soon as a DDoS attack is detected, a heuristic algorithm identifies aggressive IP addresses, and the controller instructs

978-1-7281-4387-3/19/$31.00 © 2019 IEEE

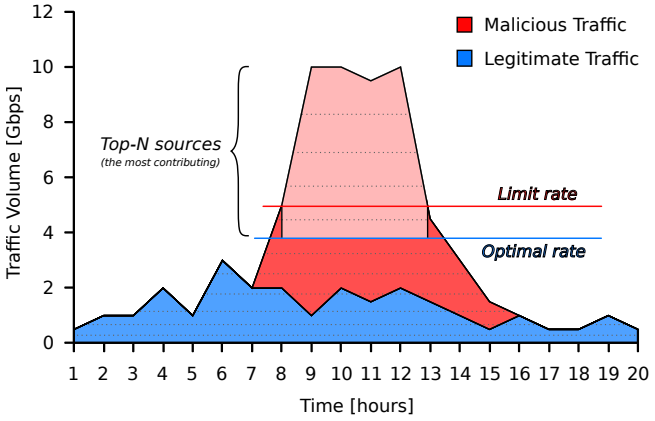

Fig. 2. Example of the mitigation principle.

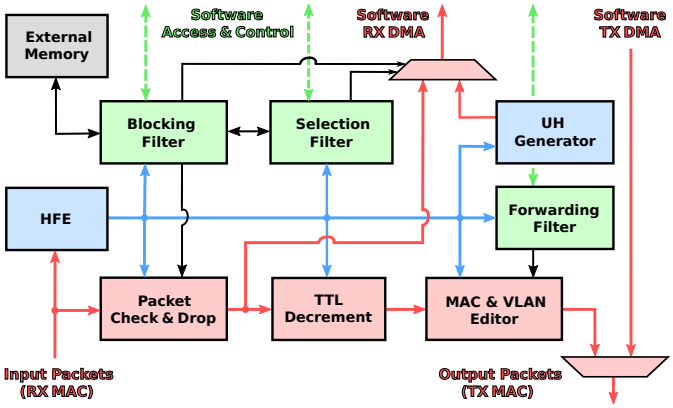

Fig. 3. FPGA firmware architecture.

the data plane to block this traffic immediately. However, the essential part is to identify the offending IP addresses correctly.

In this work, we focus on the specific family of volumetric DDoS attacks called reflective amplification attacks [3]. In this case, the malicious packets hitting a victim are reflected from a public server (e.g., an open DNS resolver) and thus contain a valid (i.e., not spoofed) source IP address of the reflector. Therefore, blocking the traffic from the particular source IP addresses appears to be effective.

Fig. 2 shows the mitigation principle. The technique aims to decrease the volume of traffic to an acceptable intensity so that it does not saturate the links of the victim server, and the target infrastructure can handle it. To this end, an operator must specify a set of rules for a subset of traffic and each protected network to define requested traffic limits. In general, such a rule consists of two parts: (1) a condition, (2) limit and optimal traffic rates. The condition specifies which packet should match the rule (e.g., protected destination IP prefix, TCP/UDP ports). The limit defines the rate of packets or bits per second that must be exceeded to start filtering the traffic targeting the protected network (matching the condition). The optimal traffic rate defines the desired rate to which the traffic should be reduced. Finally, as soon as the limit is exceeded, the algorithm filters traffic coming from the *top-N* most contributing sources, i.e., it selectively blocks source IP addresses producing the majority of malicious traffic, while the number of *top-N* blocked IP addresses is selected so that the desired optimal traffic rate is achieved.

III. Firmware Design

The FPGA firmware architecture consists of nine major building blocks. Fig. 3 depicts a view of the architecture and illustrates the operations performed for each incoming packet. The structure is organized around three main paths: the packet data path (red), packet headers and metadata (blue), and the software access and control path (green).

Each incoming packet is first parsed, and flow identification fields are extracted in *HFE (Header Field Extractor)*. The desired interesting packet headers and metadata are formed into a predefined format (Unified Header) in *UH generator*. Then, according to the software-defined rules, *Selection Filter* decides in which form and which packets are delivered to the software part of the system for further analysis. Data can be forwarded either in the form of whole packets or as Unified Headers. In case of UHs, only a few bytes for each packet are transferred to software, thus reducing the PCIe utilization and CPU load, because the packet is fully parsed in hardware, and only necessary data are transferred.

Malicious traffic coming from the particular offending IP addresses is blocked using *Blocking Filter*, as requested by the software controller. Blocking filter utilizes an exact match cuckoo hashing based table. It is realized using external on-board QDR memories and has a total capacity of over 250 000 rules. The filter also maintains packets and bytes statistics of each blocked IP address and provides them to the controller. The individual packets marked by the filter are dropped in *Packet Check & Drop* component, which also performs invalid packets checks (invalid checksums, etc.). To correctly deliver the cleansed traffic back to a target network, the mitigation device has to support some of the router forwarding capabilities. Thus, *TTL Decrement* ensures the decrement of TTL, or Hop Limit for IPv6 respectively, and drops the packet if it zeros out. Finally, *Forwarding Filter* implements an LPM (Longest Prefix Match) IP lookup algorithm to determine the next hop, and *MAC & VLAN Editor* replaces the destination MAC address and optionally the output VLAN tag.

IV. P4 Implementation

We first created a pure hand-optimized VHDL implementation. Later, we also ported the architecture into a high-level description in P4 using match-action tables.

The P4 (Programming Protocol-independent Packet Processors) [4], [5] is an open-source, high-level and platform-agnostic language. Its purpose is to provide a way to define packet processing functionality of network devices, paying attention to reconfigurability in the field, protocol, and target platform independence. P4 allows to define five aspects of packet processing: (1) *Header formats*, (2) *Packet parser*, (3) *Table specification*, (4) *Action specification* and (5) *Control*

TABLE I
ARCHITECTURE HW RESOURCES UTILIZATION RESULTS.

Implementation	LUTs	REGs	BRAM	Frequency
Virtex US+ (VHDL)	43 068	62 277	3 780 kb	200.3 MHz
Virtex US+ (P4)	117 906	162 484	2 502 kb	202.9 MHz
Stratix 10 (P4)	105 333	233 456	2 601 kb	189.8 MHz

program. We provide a more detailed description of the mapping between our architecture and P4 match-action constructs. In our case, we are using the compiler toolkit from $P4_{14}$ (the previous release) to VHDL [6].

In P4, we implemented *HFE* as the packet parser and header definitions. It describes the finite state machine used to traverse packet headers from start to end, extracting field values as it goes. Also UH structure is a kind of header, and we defined it as a P4 program metadata item. Then, we easily described *UH generator* as a P4 action which fills the UH metadata with desired parsed headers. Each filter component, namely *Selection Filter*, *Forwarding Filter* and *Blocking Filter*, implements a single match-action table that have several user-defined actions. *Selection Filter* uses ternary match, *Forwarding Filter* uses prefix match, and *Blocking Filter* utilizes exact match. Moreover, *Blocking Filter* works with stateful counters to maintain the statistics. Unfortunately, the compiler toolkit does not currently support mapping counters into external memory. Thus, the collection of statistics stayed implemented outside of the P4 pipeline. Finally, *Packet Check & Drop*, *TTL Decrement* and *MAC & VLAN Editor* map into user actions using elementary instructions to manipulate packet headers.

V. EXPERIMENTAL RESULTS

To quantify requirements of the architecture in terms of hardware resources, we synthetized P4 code and also the pure hand-optimized VHDL implementation for two different FPGAs, i.e., Xilinx Virtex UltraScale+ (model XCVU7P) and Intel Stratix 10 (model 1SG280HU). Tab. I shows the chip occupancy and frequency for both implementations and platforms. In contrast to the original VHDL implementation, designs generated from P4 occupy more resources. Due to a higher level of abstraction, it is 2-3× more in case of lookup tables and 3-4× more in case of registers. On the other hand, because of high-throughput optimizations, the original implementation uses about 50 % more block RAM.

Fig. 4 shows the overall throughput using Virtex UltraScale+ platform. The P4-generated design sustains very high throughput (almost 90-95 Gbps). However, the optimized VHDL implementation achieves nearly wire-speed throughput (99.9 Gbps) for almost all frame lengths. There appears only small inefficiency for the shortest (64-96 B) frames. Although the P4 results in more hardware resources and lower throughput, the difference between P4 and hand-optimized code is small enough to justify the flexibility of P4.

VI. CONCLUSION

We have designed an FPGA-accelerated device and DDoS mitigation technique to overcome reflective amplification

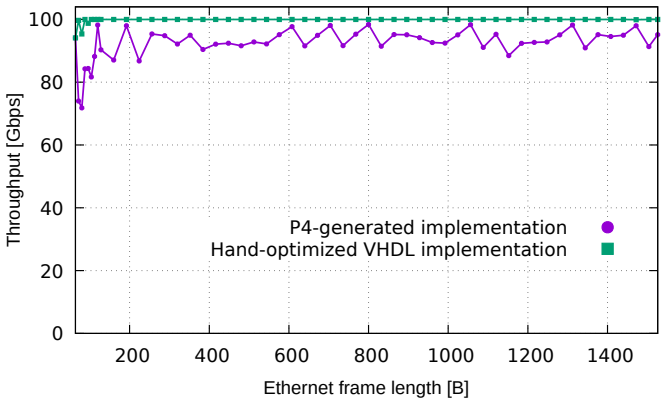

Fig. 4. Virtex US+ implementation throughput.

DDoS attacks. It is currently deployed in our infrastructure. Its purpose is to filter malicious traffic at high-speeds in the backbone network to prevent the attacks from saturating victim's links. It blocks individual source IP addresses based on their volumetric contribution to the attack. However, considering the high-level P4 implementation, the device offers an interface to implement other mitigation algorithms. For example, we develop a heuristic approach to mitigate TCP SYN flood attacks, and our future work is to further extend mitigation algorithms to allow for their flexible utilization according to the current attack surface. In future work, we will also describe the design of the software controller and our experience with the device deployment. We will further evaluate it and compare its performance and accuracy to other existing solutions.

REFERENCES

[1] O. Kupreev, E. Badovskaya, and A. Gutnikov, "DDoS attacks in Q2 2019," Kaspersky Lab, Tech. Rep., Aug. 2019, Accessed: 2019-08-20. [Online]. Available: https://securelist.com/ddos-report-q2-2019/91934/

[2] Cisco Systems, Inc., "Cisco Visual Networking Index: Forecast and Trends, 2017–2022 White Paper," Tech. Rep., Jan. 2017, Accessed: 2019-08-20. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.pdf

[3] V. Paxson, "An analysis of using reflectors for distributed denial-of-service attacks," *SIGCOMM Comput. Commun. Rev.*, vol. 31, no. 3, pp. 38–47, Jul. 2001. [Online]. Available: http://doi.acm.org/10.1145/505659.505664

[4] 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. [Online]. Available: http://doi.acm.org/10.1145/2656877.2656890

[5] P4 Language Consortium, "P4," 2019, Accessed: 2019-08-20. [Online]. Available: http://p4.org/

[6] P. Benáček, "Generation of high-speed network device from high-level description," Ph.D. dissertation, Czech Technical University in Prague, Faculty of Information Technology, Thakurova 9, 160 00, Prague 6, 3 2016.