Highlighting the Gap Between Expected and Actual Behavior in P4-enabled Networks

Nicholas Gray, Alexej Grigorjew, Tobias Hosssfeld
University of Würzburg, Germany
Apoorv Shukla, Thomas Zinner
TU Berlin, Germany

***Abstract*—Modern networks increasingly rely on Software-defined Networking (SDN) and Network Function Virtualization (NFV) to augment their flexibility in high load scenarios. To further enhance the performance, a part of the functionality is often offloaded to forwarding devices, which are used as hardware accelerators and are configured by high level programming languages such as P4. However, hardware vendors use sophisticated technologies to implement these standards, which need to be understood by the programmer to avoid unintended behavior. In this demonstration we highlight the severe consequences of only relying on the network programming language when ignoring the device-specific limitations. We show this by the example of a Denial of Service attack against a P4-enabled SmartNIC. Finally, we discuss possible mitigations to this attack and stress the importance of an overall understanding of the entire system.**

I. Introduction

Modern networks often rely on computation-intensive Network Functions (NFs) during their daily operation, such as Intrusion Detection Systems (IDS) and sateful packet filter firewalls. At the same time, new concepts emerge, e.g. Software-Defined Networking (SDN) [1], allowing to programm the behavior of network devices via OpenFlow [2] and P4 [3]. As performance is a critical aspect to most networks, these concepts are commonly used to offload some functionality to hardware as shown in [4]. However, the underlying hardware implementation adds a new layer of complexity to the network which needs to be verified and understood.

As hardware vendors attempt to implement behavior from high-level programming languages into their devices and process packets at line-rate, they need to embed the respective operations into their specific architecture and map data to available resources. Therefore, they often rely on sophisticated technologies to achieve gigabit performance. Content-Addressable Memory (CAM) is typically used in the Forwarding Information Base of regular Ethernet switches. Ternary Content-Addressable Memory (TCAM) is an extension typically applied in routers and OpenFlow-enabled switches that allows the use of wildcard bits to retrieve information based on only a part of its identifier. Caches are used to store and quickly apply forwarding behavior for packets with specific headers, e.g. in Multicore-Systems on a Chip (Multi-SoCs). The capacities of these approaches, however, are always limited, and the programmer should be aware of their implications.

These implications include vulnerabilities and performance issues when exposed to specific traffic patterns. To guarantee a secure operation of the network with good performance, the utilized software and hardware has to be resilient and tested against edge traffic patterns. An effort to identify workloads that trigger slow execution paths for non-proprietary softwarized NFs has recently been conducted in [5]. Yet, a related approach for proprietary software and hardware NFs is still missing.

In the remainder of this work, we demonstrate such a mismatch with severe implications for Multi-SoC based Netronome Agilio CX SmartNICs [6], [7]. These devices offer flexible programming via P4 and are already used by major companies such as DELL and Intel for NF virtualization ready server solutions [8], [9]. In addition, Microsoft Azure has built its cloud network on host-based SDN technologies while using FPGA-based SmartNICs as hardware accelerators [10]. In the following, the limitations of the caching strategy of this 10 GbE SmartNIC are demonstrated by performing a Denial of Service (DoS) attack with a mere 100 Mbps data rate against a stateless packet filter firewall implemented in P4. Afterwards, possible countermeasures for this particular problem are proposed, and the general implications of control- and data-plane mismatches are summarized.

II. Demonstration

In this demonstration, a firewall offloading scenario is investigated in which a 10 GbE SmartNIC is running a stateless packet filter firewall programmed in P4[1]. As shown in Figure 1, a Spirent traffic generator is used as traffic source and sink. Therefore, one port generates both the legitimate and attacking traffic, while the other port operates as the receiving end of a web server. The firewall forwards all packets to the Spirent test center which belong to the web server, based on their destination IP address, the layer 4 protocol, and the destination port, while dropping all other packets. This is achieved by two simple match-action entries, as shown in Table I. Note that the presented vulnerability is not specific to this particular program or configuration, as the caching behavior is currently a fixed, built-in function of the device and not under the control of the programmer.

In the first part of the demonstration, the general functionality of the SmartNIC is verified. Therefore, Spirent generates 8 Gbps TCP traffic mix whose destination IP and port match the allowed configuration, as depicted in Table I. Thereby, no

[1]https://github.com/lsinfo3/2018-P4-Firewall

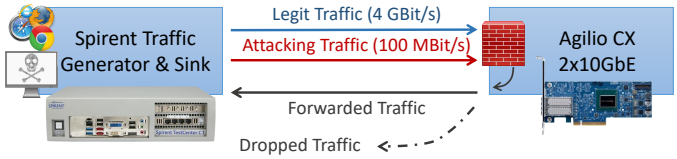

Fig. 1: Experimental setup. The SmartNIC on the right runs a P4-enabled stateless firewall, forwarding traffic back to the Spirent test center.

TABLE I: Firewall ACL configuration.

Match	Action	Prio.
DstIp: <IP>, Prot: TCP, DstPort: 80	Allow	10
Any	Deny	9

packet loss is observed, as the Netronome SmartNIC is able to handle this homogeneous workload.

In the second part, the vulnerability that causes packet loss is demonstrated. Thereby, the traffic generator starts by sending the attacking 100 Mbps traffic mix only, and adds 4 Gbps of a common web traffic pattern to the workload after 30 seconds. The attacker's stream generates packets with varying source IP addresses and source TCP ports while keeping the destination fixed, as required by Table I. Figure 2 displays the rate of sent and received packets during each second of the evaluation.

Up to 18 seconds of the experiment's duration, the received data rate matches the sending rate, hence the device forwards all incoming packets correctly. After 18 seconds, severe packet loss is observed for the attacker's traffic. At this point, all two million available cache entries are occupied by previous packets. This behavior persists after adding the legitimate traffic to the workload at 30 seconds, effectively blocking most communication traversing this SmartNIC, despite previously proving that it can reliably handle twice the amount of throughput. After 38 seconds, the malicious entries in the cache expire and are replaced by legitimate addresses, and most packets can be forwarded as intended. However, it should be noted that this stems from the homogeneous traffic pattern, consisting of a limited number of source IP addresses which is rather unlikely for data centers and short-lived web flows. Hence, a more realistic scenario is more likely to be affected by packet drops, as the changing flows would even augment the limitations of the underlying caching strategy.

Unfortunately, the algorithm for generating the key of each cache entry cannot be adapted by the underlying P4 program and always takes the bits into account which correspond to the source IP and port of a regular TCP/IP packet.

III. Discussion & Conclusion

The demonstration shows that it is easy to compromise security applications deployed in SmartNICs by exploiting their caching behavior, if their device-specific implementation is not considered. Hence, not only the application itself but also the underlying technology has to be taken into account to guarantee a secure operation of productive networks. To

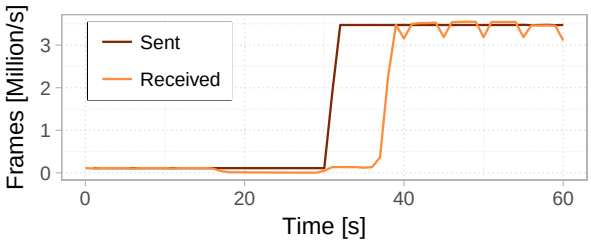

Fig. 2: Sent and received frame counts during the experiment.

mitigate such DoS attacks, one possibility is to allow the definition of header fields on which the cache is based. This would allow to shift the caching process towards fields which are unaccessible for an attacker.

For future work, it needs to be considered how a Secure Development Life-cycle (SDL) for programmable networking devices can be established. This includes the development of new tools, which take the device-specific limitations into account, as well as adaptations to integration tests and formal verification procedures.

To conclude, these solutions introduce an additional layer of complexity, as a deeper understanding of the underlying technology and its traits is required. Therefore, the capabilities of these devices should be stated clearly and kept in mind when using a high level language to program their behavior. At last, the implementation of the device should strive to follow the specifications as closely as possible to minimize the overhead introduced by this new layer of complexity.

References

[1] D. Kreutz *et al.*, "Software-Defined Networking: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 103, no. 1, pp. 14–76, 2015.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[3] P. Bosshart *et al.*, "P4: Programming Protocol-Independent Packet Processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.

[4] A. Sapio, I. Abdelaziz, A. Aldilaijan, M. Canini, and P. Kalnis, "In-network computation is a dumb idea whose time has come," in *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*. ACM, 2017, pp. 150–156.

[5] L. Pedrosa, R. Iyer, A. Zaostrovnykh, J. Fietz, and K. Argyraki, "Automated synthesis of adversarial workloads for network functions," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. ACM, 2018, pp. 372–385.

[6] "Netronome. Open vSwitch Offload and Acceleration with Agilio CX SmartNICs," https://www.netronome.com/media/redactor_files/WP_OVS_Benchmarking.pdf, accessed: 2018-09-17.

[7] "Agilio CX SmartNICs," https://www.netronome.com/media/documents/PB_Agilio_CX_2x10GbE.pdf, accessed: 2018-09-17.

[8] "Netronome Collaborates with Dell EMC OEM Solutions to Deliver Turnkey NFV Server Solution," https://www.netronome.com/press-releases/netronome-collaborates-dell-emc-oem-solutions-deliver-turnkey-nfv-server-solution/, accessed: 2018-09-17.

[9] "Netronome, Advantech and Spirent Demonstrate Solution for SDN-Enabled Appliances and Servers," https://www.netronome.com/press-releases/netronome-advantech-and-spirent-demonstrate-solution-for-sdn-enabled-appliances-and-servers/, accessed: 2018-09-17.

[10] D. Firestone *et al.*, "Azure Accelerated Networking: SmartNICs in the Public Cloud," in *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*. USENIX Association, 2018.