The Case for Ultra High Speed Portable Network Security Filters

Alessandro Rivitti[1,2], Angelo Tulumello[2] and Giuseppe Bianchi[1]

[1]*University of Rome Tor Vergata, Rome, Italy*
[2]*Consorzio Nazionale Interuniversitario per le Telecomunicazioni*

Abstract

Country-wide infrastructures and large-scale data centers demand protection against high-end cyber threats, potentially including nation-state attackers and vulnerabilities deployed within the infrastructure itself through supply chain attacks. In such a scenario, a compelling defensive strategy consists of running protection measures on *different hardware* than the one supporting the applications being protected. While this is relatively simple for protection mechanisms that can be implemented in software, the challenge lies in achieving this goal when deploying protection means capable of being dynamically migrated and run at wire-speed over network links at 100 Gbps and more.

In this direction, this paper proposes the notion of Portable High-Speed Network Security Filters. These are packet/flow-level filters that can be written once with a single programming abstraction, migrated inside different network vantage points, including Smart NICs and even the newly emerging Small Form-factor Pluggable switch Ports (SFPs), and capable of providing hardware-grade performance and wire-speed operation.

To implement and deploy such filters, we propose to combine the portability of eBPF software code with the programmability of FPGA hardware. Moreover, filters can be deployed with no hardware programming effort, by further employing our novel technology called eHDL, which allows the compilation of eBPF tasks into highly efficient VHDL hardware pipelines. The feasibility of our proposed approach is concretely showcased over 100 Gbps network links, by demonstrating the seamless porting of the filtering part of the well-known Suricata Intrusion Detection System, a Firewall and a DDoS mitigation Network Function to an Alveo U50 SmartNIC.

Keywords

eBPF, FPGA, Network Security, IDS

1. Introduction

The increasing volume and diversity of internet traffic, coupled with the ever-evolving cyber threat landscape, demand advanced and scalable network defense frameworks. When defending nationwide network infrastructures or high-end enterprise data centers, crucial factors for effective defense include the ability to real-time manage very high traffic volumes (hundreds of Gbps) for prompt detection and response to attacks.

Furthermore, evolving cyber threats require agile defense. Direct programming of packet processing tasks on programmable hardware like FPGA offers flexibility and hardware-grade performance. However, programmability alone may not suffice; equally vital is the ease and affordability of implementing adjustments and updates to protection mechanisms by the opera-

ITASEC 2024: The Italian Conference on CyberSecurity

tional team. This becomes especially critical in the face of sudden attacks, where relying on product developers for timely updates may not be practical.

In addition to the above considerations, when striving to protect against high-end cyber threats, including the possibility of nation-state attackers exploiting vulnerabilities through supply chain attacks, a new critical aspect arises: *determining the physical deployment of protection measures*. There is a growing concern about the potential presence of untrusted hardware devices, with a community sentiment [1, 2, 3] suggesting that vulnerabilities might even be embedded in third-party commercial off-the-shelf hardware components. Consequently, greater effort must be pursued into the separation between the hardware running protection measures from the hardware supporting the applications being protected. This would ensure the avoidance of single points of failure that could occur if both defense and the protected system operate on the same hardware platform.

Right now, this challenge, especially when dealing with high performance and scalable approaches, has only been marginally addressed. Most efforts to fight malware have in fact focused on two main areas: Network-wide protection via perimetral defenses and Network Intrusion Detection Systems (NIDS) and end-point protection via specialized security software such as Endpoint Detection and Response (EDR) or their extension into Extended Detection and Response (XDR), which involve software agents installed in end hosts, servers and network nodes.

We posit that the current disaggregation trend [4, 5] we are witnessing in networked systems is a great opportunity not only to improve network performance, but also to improve security. The emergence of technologies such as Smart Network Interface Cards (Smart NICs) or even Smart Small Form-factor Pluggable switch Ports (SFPs) [6, 7, 8] would in principle permit to deploy network intelligence at the microscopic level of even a single switch port, thus improving defensive security measures via a more capillary and diversified spread of protection means across the entire network infrastructure.

In this strategic direction, our specific contribution revolves around the promotion of what we refer to as *Portable High-Speed Network Security Filters*. Conceptually, the idea is simple: design network traffic filters which can be migrated wherever needed inside the network, and deployed across different physically distinct hardware, including smart NICs and smart SFPs. To make such a vision practical, the crucial question to ask is: which technology choices are appropriate to enable seamless portability across software components and programmable hardware? We explicitly target programmable hardware, as high-performance solutions need fast stream processing of traffic and quick response times, and the challenge is how to retain the flexibility and programmability to keep up with the constantly evolving landscape of cyber threats and attacks, meanwhile enabling a forward-looking near-Tbps-level efficiency of traffic processing directly enforced via hardware processing pipelines.

In this paper, we advocate for the combination of two standard technologies (eBPF and FPGA) via a third novel technology, which we have recently proposed under the name eHDL. While eBPF takes the role of portable programming language for our filters, and FPGA provides the programmable hardware support, eHDL permits to seamlessly "compile" eBPF network security filtering tasks into *extremely efficient* VHDL hardware pipelines which can be operated at hundreds of Gbps and can hence be deployed to strategic points within the network infrastructure - smart NICs and SFPs being natural targets.

Other than highlighting the potential benefits of this approach, we concretely demonstrate its feasibility through test cases operating at 100 Gbps speed, by showing the seamless porting of two eBPF-based network functions, the well known Suricata IDS and a custom firewall, within an Alveo U50 SmartNIC.

2. Motivation, Technical Challenges and Use Cases

There are several "classical" reasons that advocate for the design of traffic filtering mechanisms capable of being deployed at various points in a network.

Improved Security. This is the main motivation for relocatable security functions, as the distribution of packet filtering and firewalling functions enables a more capillary protection approach. It permits to more easily and naturally manage and isolate security domains, and implement more advanced forms of adaptive security posture, where security measures can be tailored to specific segments of the network. Moreover, it ensures that security measures are not centralized in a single point but are instead spread across the network, offering a more robust defense against potential threats, and a greater resilience and robustness to failures. Additionally, it facilitates the detection and prevention of lateral movements, which are typically challenging to counteract using solely centralized Network Intrusion Detection Systems (NIDS) [9, 10, 11].

Performance. Another traditional and well known motivation revolves around performance. Portability and distribution of security tasks, accomplished by placing packet filtering and firewall functions closer to the source or destination of the traffic, can indeed help reducing latency, improve scalability, and optimize resource utilization.

Hardware Diversity. In addition to all this, we posit that a third compelling and perhaps more fresh motivation resides in the opportunity of diversifying the platforms which run the defensive mechanisms from the hardware that runs the potential attacker's target. This is especially important in scenarios where hardware trustworthiness many not be anymore guaranteed [1], e.g. because of potential supply chain attacks [2, 3] by nation state actors. Placing, for instance, traffic filters over a smart NIC, instead of deploying it as an agent running on the end host, provides a diversified and more secure architecture by isolating critical security functions from host vulnerabilities, hence mitigating the risk of attackers escalating privileges on the host. Indeed, Smart NICs operate with their own dedicated firmware and processing capabilities, so that the control of the end host by an attacker is not yet sufficient to disable the defense deployed in the NIC. We argue that Hardware Isolation of security tasks is a well established concept in Hardware Security contexts [12, 13, 14, 15] however the Network Security domain lacks this aspect.

2.1. Technical Challenges

On its own, the dynamic placement of security functions across the network wouldn't pose any specific or unique challenges when handled through software implementations. However, the scenario changes drastically when the objective is to create programmable traffic filtering and

processing tasks designed to operate at wire speed over very high-speed links, typically in the order of hundreds of Gbps.

The natural solution would be to leverage FPGA technology, which is today deployed over several Network Interface Cards (i.e., smart NICs) and which we expect to be included in the future also within smart Small Form-factor Pluggable switch Ports (SFPs). FPGA-based NICs offer excellent performance and high flexibility, allowing programmers to define a wide range of functions within available hardware resources. Unlike other NIC accelerators like network processing ASICs or many-core System-on-Chip SmartNICs, FPGA NICs provide added flexibility to support diverse accelerators for various applications. However, programming FPGAs is challenging and often requires a dedicated team of hardware specialists, complicating integration with software and operating system development.

We address such challenge by designing our portable hardware filters as detailed in section 3, i.e. using the eBPF programming language for coding the filters themselves, and our recently proposed eHDL technology [16] for compiling eBPF programs into VHDL .

2.2. Potential Use Cases

With no pretense of completeness, we discuss hereafter a few examples where we envision Portable Filters placement and argue how this can bring security improvements.

- Switch Small Form-factor Pluggable (SFP) Ports: A Portable Filter could be placed inside the SFP of a switch allowing for the monitoring of all traffic passing through that port. This is particularly effective for detecting malicious traffic or unauthorized data exfiltration attempts on specific network segments.
- Between the Firewall and Network Switch: Placing a Portable Filter between the firewall and the network switch can provide an additional layer of security. This placement can be used to further scrutinize traffic that has been allowed by the firewall, ensuring that threats do not slip through cracks due to firewall misconfiguration or zero-day vulnerabilities. This scenario of course require very clever and complex functions to be handled by the Portable Filter itself.
- Directly Connected to Servers or Critical Endpoints: For high-value servers or critical infrastructure components, Portable Filters can be placed directly before these devices. This setup ensures that any data sent to or from these critical assets is inspected, possibly tailoring the Filter to the specific needs of the device.
- In Branch Office Connections: In a distributed network architecture, Portable Filters can be placed at the entry/exit points of branch office connections to the main network. This can help in enforcing consistent security policies across the organization and in monitoring traffic for potential threats specific to remote locations.
- Virtualized Environments: In networks that heavily utilize virtualization, Portable Filters can be deployed between nodes that deploy several virtualized network segments. This allows for the monitoring and control of traffic between virtual machines, which is crucial in a cloud or virtualized data center environment.
- VPN Gateways: For networks that utilize Virtual Private Networks (VPNs) for remote access, placing Portable Filters at the VPN gateways can help in inspecting encrypted

traffic once it is decrypted at the gateway. This ensures that malicious traffic from remote devices is caught before it enters the main network.
- Old Untrusted Hardware: A Portable Filter could be placed at the edge of possibly vulnerable hardware devices that might have been infected due to possibly outdated software or simply put hardware devices that we're unsure whether they can be exposed to the internet or not.
- Hardware Trojans/ Hardware Bugs: a malicious activity can be detected by the Portable Filters that are directly connected to Hardware devices that potentially are vulnerable to hardware exploitations (Spectre, Meltdown) or simply we're not sure about third-party commercial off the shelf (COTS) hardware we want to use and prefer a Portable Filter implementation rather than developing our own custom hardware.
- IoT Gateway: an IoT environment has to enforce security over several different devices. A Portable Filter implementing a carefully deployed IDS for IoT devices [17] can prove viable securing the whole IoT network cascaded from this device.

In the above scenarios, we envision Portable Filters as a plug-and-play way of deploying security across different network vantage points, without the need for comprehensive changes to the existing networking infrastructure.

3. Tecnological Enablers and Proposed Solution

This section aims to prove the feasibility of hardware-based portable traffic filters. We first describe the two crucial catalysts for our Portable Filters, and specifically the eBPF programming language and the FPGA hardware technology. We then describe eHDL, an innovative approach proposed in our recent work [16]. As detailed later, eHDL is, in short, an eBPF to VHDL compiler which permits to produce, with no human effort in terms of hardware programming, very efficient hardware pipelines from eBPF packet and traffic processing scripts. By merging the flexibility of eBPF with the performance effectiveness of FPGA, without requiring any specific HW programming skill from the designer, eHDL appears to us as a valuable and very effective technology for deploying Portable Filters across the entire networking stack. Moreover, it is worth to remark that our approach builds upon fully established technologies, ensuring backward compatibility and reusability, eliminating the need to develop new hardware devices or new programming languages. Indeed, as discussed in Section 4, we show the seamless porting of the unmodified eBPF filtering portion of Suricata, a well known and widely used IDS, into an FPGA Smart NIC.

3.1. eBPF

The original Berkeley Packet Filter (BPF) was introduced in 1992 to efficiently filter packets within the kernel space before capturing them in a user space application. In recent years, the extended Berkeley Packet Filter (eBPF) has emerged as an even more powerful and flexible tool for programming the Linux kernel without altering its source code or loading kernel modules. Its versatility spans various domains, including networking, security, and performance monitoring, making it well-suited for our objectives. We require a language that allows network

engineers to define precise actions on packet streams, and eBPF is an ideal choice for two reasons: i) its widespread usage and familiarity, and ii) its XDP hook is specifically tailored for networking, avoiding the loss of generality associated with more general-purpose languages. XDP programs operate independently of the broader network processing stack, preemptively addressing critical security and management tasks to regulate traffic before it reaches the host system. In production environments, XDP is employed for packet filtering (e.g., Suricata), load balancing (e.g., Katran), and DDoS detection/mitigation (e.g., Cloudflare's L4Drop).

Examples of sophisticated stateful Network Security Functions suited to be implemented with eBPF/XDP include:

- **Network Intrusion Detection and Prevention Systems**: With the enhanced visibility of the eBPF framework, the programmer can create advanced packet filtering rules that go beyond traditional IP/port filtering, allowing for deep packet inspection (DPI) and analysis. In this way, it can apply complex logic and rules to detect malicious patterns making on-the-fly decisions to either drop, forward, or modify traffic [18][19][20].
- **DDoS Mitigation**: Efficiently absorbing large DDoS attacks is becoming more and more important in recent years, given the rapid growth in access networks speeds. eBPF and XDP help solve the performance requirements for mitigating such attacks by intercepting and discarding unwanted traffic before it reaches the applications [21].
- **Firewalls**: Originally, software implementations of firewalls used NETFILTER/iptables as the data plane choice. eBPF is a suitable option for efficiently classifying packets and executing complex network policies with minimal performance overhead [22].
- **Security Policies for CNIs (Cilium)**: Container Network Interfaces (CNIs) are network plugins for K8s, enabling the networking between pods in a K8s deployment. Recently, many popular CNIs like Cilium[23] and Calico[24] have transitioned from utilizing NETFILTER/iptables for the implementation of network functions to adopting eBPF, aiming for enhanced scalability. An important feature of a CNI is the execution of Network Security policies defined by the user, which is implemented using the eBPF framework as well [25].

For these reasons, eBPF was selected as the enabling technology for programmability in our Portable Filters deployment.

3.2. FPGA

We now turn our attention to the numerous advantages that position FPGA hardware as the second critical enabler for the deployment of Portable Filters. The rationale is threefold, as outlined below:

- i) **Performance**: FPGA performances for filtering workloads are unmatched by software implementations of the same functions;
- ii) **Programmability**: compared to ASIC counterparts, FPGAs trade off performance with programmability;

- iii) **Portability**: We argue that FPGAs are being deployed in several points of the computing stack, thanks to the new trend of heterogeneous computing. In this context, FPGAs prove to be enablers for the physical separation we need in our Portable Filters.

FPGAs offer exceptional performance in packet processing, boasting high throughput (∼100Gbps) and low latency (∼100ns). Despite the traditionally high engineering costs linked to low-level languages like VHDL and Verilog, the mainstream adoption of FPGAs has surged. This shift is attributed to the rise of High-Level Synthesis (HLS) tools like Vitis HLS and Chisel, which simplify FPGA design by supporting higher-level languages such as Python and C++.

Moreover, in recent years FPGA vendors are targeting towards the future approach of heterogeneous computing where general purpose CPUs alone can't handle the growing need for specialized hardware. To try and keep on scaling computing performances in the post Moore era, a new approach is being leveraged. The core concept involves distributing different computing tasks across distinct hardware modules; for instance, neural network inference tasks are expedited by the Neural Engines. This approach is exemplified by System on Chips (SoCs) that integrate segments of FPGA fabric with general-purpose processors, such as ARM or RISCV, within the same chip. A case in point is the AMD-Xilinx Versal ACAP, where FPGA resources, multiple "Scalar Processors", and Neural Engines dedicated to neural network inference coexist on a single chip. Other examples are Microchip Polarfire SoC[26], Intel Agilex SoC[27]. Moreover, interest in the chiplet market is rising for several reasons, including enhanced chip performance and improved production yields. The fundamental idea here is to manufacture specialized hardware chips that can later be modularly linked with each other. It is noteworthy within this context that the FPGA market is expanding and increasingly capturing the attention of leading chip manufacturers, such as AMD-Xilinx and Intel-Altera.

In this heterogeneous landscape, we envision the deployment of reconfigurable hardware in a variety of devices. Expanding on this vision, we identify four scenarios where a small FPGA segment could be integrated into the computing stack, aligning with our Portable Filter deployment strategy. As elaborated in Section 4.2, Portable Filter implementations require a modest amount of FPGA resources, thereby facilitating deployment in smaller form factor FPGA devices.

- A possible deployment scenario could be embedding a Portable Filter directly within the Small Form-factor Pluggable (SFP) module of a switch. This allows for comprehensive traffic monitoring and the detection of malicious activities as all data passing through that particular switch port is scrutinized in real-time.
- Portable Filters can as well be strategically positioned at the network interface card (NIC) level of individual devices. By integrating security measures at the NIC, every communication originating or terminating from the device can be filtered.
- Inside a server that uses a System on a Chip (SoC), we can place a Portable Filter right within the FPGA portion of the SoC itself alongside the CPU. This way, it's in the same chip as the computing component but still kept separate, boosting security by keeping different functions divided.
- Another pivotal placement for Portable Filters is within the router infrastructure. By installing these secure devices at key junctures in the routing process, network adminis-

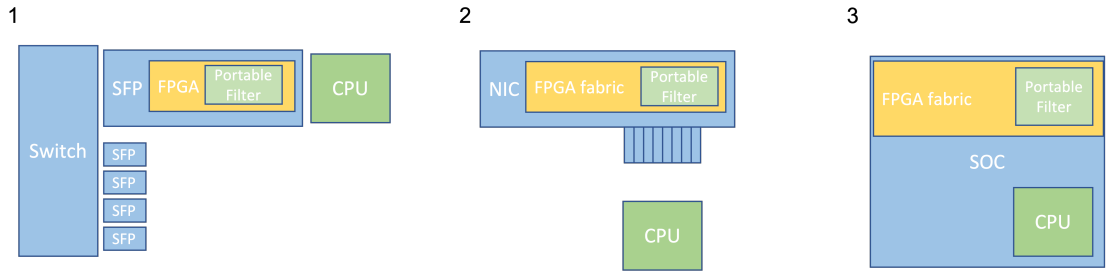

Figure 1: Possible placement of Portable Filters

trators gain the ability to scrutinize and filter incoming and outgoing traffic, effectively thwarting potential threats before they reach critical network components.

In figure 1, we illustrate the first three examples we discussed (the fourth would have been represented similarly) to show the physical separation we envision. It is worth to mention that the Portable Filters applicability is not limited to the four examples presented above. Instead, we believe that forthcoming technological advancements will facilitate additional placement options throughout the entire networking stack.

Clearly, the complexity of the Portable Filter function is closely tied to the specific FPGA module adopted. For example, implementing a complete Network Intrusion Detection System (NIDS) might demand substantial FPGA resources, making it suitable for inclusion in a SmartNIC. Conversely, a simple Firewall or Access Control List could be easily accommodated within small form factor FPGA boards, such as those found in a System on Chip (SoC), potentially allowing for multiple implementations within the same SoC.

3.3. eHDL

Our main contribution that paves the way for simple implementation of Portable Filters is eHDL [16].

eHDL is a compiler that takes unmodified eBPF/XDP programs and deploys custom FPGA designs. The core of the eHDL design is a packet processing pipeline written in low-level VHDL code that is tailored to the eBPF program. In this way we provide careful resource usage while maintaining high performances. The eHDL pipeline is not bounded to a single FPGA SmartNIC and is customizable enough to be placed in boards from different FPGA vendors.

The concept of Portable Filters can be enabled by eHDL. We leverage the needed programmability given by eBPF needed to keep up with increasing and evolving threats while utilizing the power of hardware implementations of Network Function. It is notable, and it will be further examined in Section 4, how implementations of simple Network Security Functions are viable with a minimal impact on hardware resources.

As FPGA devices are being deployed in more and more portions of the computing stack, our vision extends to an eHDL pipeline capable of executing real-time stream packet processing across several devices.

eHDL proves viable to implement Portable Filters due to the following reasons:

- Programmability: Leveraging eBPF for networking functions ensures that our approach remains flexible and adaptable.
- High Performance and Low Power Consumption: eHDL pipelines are designed to operate at line-rate, maximizing efficiency and minimizing energy usage.
- Portability: The increasing integration of FPGA fabric across various computing layers defines our system's adaptability.
- Physical Decoupling: By isolating the Portable Filter within the FPGA fabric, we achieve a clear separation from traditional hardware devices, that is typically deployed on x86 server CPUs for both security and computational tasks.

4. Experimental POCs

To validate the practicality of Portable Filtering, in which physical segregation between security functions from computing functions is enforced, we conducted three test cases employing a Firewall, an Intrusion Detection System (IDS), and a DDoS Detection and Mitigation Network Function. All of the three Network Functions have been developed using the eBPF language.

Our experimental setup includes three components: (i) an AMD-Xilinx SmartNIC Alveo u50, which serves as the Portable Filter in our network architecture; (ii) a 100Gbps Mellanox ConnectX-5 NIC equipped with a DPDK-based traffic generator, capable of achieving line-rate traffic with 64B packets, translating to 148 million packets per second (Mpps); and (iii) two distinct x86 computers that host these NICs, designated as the Attacker machine and the Computing Machine for the purposes of our study.

The two NICs are directly connected with a 100Gbps link. The Portable Filter is hosted in the Computing Machine while the Mellanox ConnectX is hosted in the Attacker machine. With this testbed, we aim to emulate the physical separation between the SmartNIC, which is solely tasked with performing security functions, and the x86 server attached to it via the PCIe bus, where network traffic is further processed. The Portable Filter acts as an intermediary between the incoming malicious traffic and the Computing Machine. In Figure 2, the testbed setup for the following proof-of-concept test cases is represented.

The three Network Functions developed in eBPF have then been implemented inside the SmartNIC using the corresponding eHDL pipeline designs, and subsequently flashed into the Alveo u50 FPGA board.

4.1. Network Functions

Suricata. Suricata [20] is an open-source IDS and Intrusion Prevention System (IPS) that is widely recognized for its efficiency and versatility. Developed by the Open Information Security Foundation (OISF), Suricata is designed to provide real-time traffic analysis and threat detection capabilities. Some portions of the packet processing code is developed using eBPF in the XDP hook which allows us to seamlessly generate its corresponding eHDL pipeline. In this paper, we target the Packet Filtering program implemented in XDP, which identifies flows by the 5-tuple and selectively blocks the traffic according to configurable rules. Unfiltered packets are efficiently directed to the host through our high-performance driver, which also implements

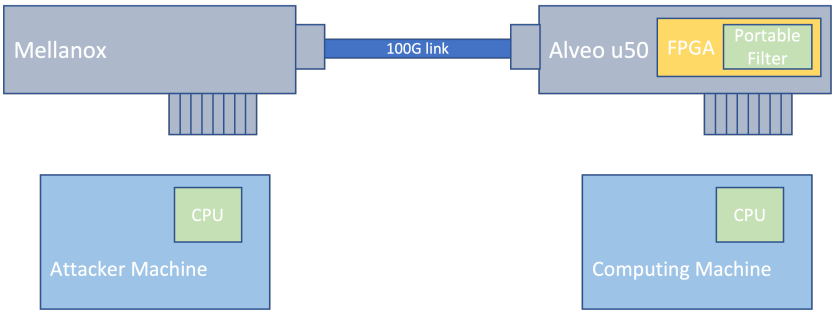

Figure 2: Testbed Representation

AF_XDP sockets for fast packet IO to user space, ensuring seamless integration with the Suricata framework for the remaining NIDS functionalities.

Firewall. The Firewall Network Function applies wildcard matching on the 5-tuple (IP source and destination addresses, Layer 4 source and destination ports, IP protocol) to classify the traffic based on configurable rules. This is achieved through the use of five eBPF hash maps, each keyed by a distinct field from the 5-tuple. Finally, the outcome from each table is represented as a bitmask, indicating the rules that were matched. These bitmasks are then combined through the use of De Bruijn sequences, resulting in the identification of the original wildcard rule that was matched. This Network function is inspired by bpf-iptables[22].

DDoS Detection and Mitigation. This Network Function is designed to perform DDoS detection and mitigation using packet filtering and rate limiting. It employs several maps to track statistics and manage blacklists for both IPv4 and IPv6 addresses, enabling it to drop or allow packets based on predefined rules. The program parses incoming packets to determine their type (IPv4 or IPv6) and applies logic to either pass, drop, or blacklist the packets based on their source IP addresses and other metrics such as the rate in Packets per Second (pps) and bits per second (bps). Additionally, it features dynamic rate limiting capabilities, adjusting thresholds in real-time to optimize network performance. This approach allows for early DDoS detection and mitigation by preemptively filtering unwanted or potentially harmful packets before they reach the application layer. This Network Function was adopted from the open-source FlowSentryX [28] project.

4.2. Performance Assessment

The three Network Functions were easily deployed in our testbed due to the programmability that eBPF offers and our eHDL FPGA integration. They performed at line rate with a modest amount of FPGA resources used, as shown in table 1. Latency values have to be intended as end-to-end values measured at the Traffic Generator side as a difference between sent and received timestamps, which means the latency value is composed by the sum of 2 times the propagation in the fiber cable (TX and RX) and the crossing of the eHDL pipeline in the Alveo u50. Resources values (Flip-Flops, LookUpTables, BlockRAMs) have been expressed as a percentage of utilization of the resources provided by the FPGA chip in the Alveo u50 board.

	Throughput	Latency	FFs	LUTs	BRAM
Suricata	100Gbps	1116ns	4.93%	7.23%	13.62%
Firewall	100Gbps	1027ns	4.95%	7.19%	13.62%
DDoS	100Gbps	1256ns	4.24%	6.28%	14.25%

Table 1
Performances and Area impact for both the Suricata and the Firewall Network Functions implemented using eHDL inside the Alveo u50 FPGA

These results demonstrate a minimal resource occupation on hardware for the implementation of Portable Filters. Although our testing was limited to a SmartNIC implementation, we are confident that adapting these filter designs to the diverse hardware stack associated with the emerging trend of heterogeneous computing including chiplets, SoCs, and small form factor FPGAs is achievable.

5. Conclusion

This paper presented our vision for Portable Filters. With this work we introduced a flexible and efficient solution to the evolving challenges of network security. Portable Filters ensure physical decoupling between computing and security functions.

Portable Filters are deployment of Network Security functions developed using the eBPF language and implemented in FPGA hardware through the use of eHDL pipeline designs.

We made efforts to prove its near future widespread adoption and provided three test cases that performed at 100Gbps at around 1 microsecond of latency.

Acknowledgments

This work was partially supported by the projects SERICS(SEcurity and RIghts In the CyberSpace - PE00000014) and RESTART (Telecommunications of the Future - PE00000001) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU.

References

[1] W. Hu, C.-H. Chang, A. Sengupta, S. Bhunia, R. Kastner, H. Li, An overview of hardware security and trust: Threats, countermeasures, and design tools, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 40 (2020) 1010–1038.

[2] M. Reed, J. F. Miller, P. Popick, Supply chain attack patterns: Framework and catalog, Office of the Deputy Assistant Secretary of Defense for Systems Engineering 2 (2014).

[3] B. Halak, Hardware Supply Chain Security: Threat Modelling, Emerging Attacks and Countermeasures, Springer, 2021.

[4] Y. Shan, W. Lin, Z. Guo, Y. Zhang, Towards a fully disaggregated and programmable data center, in: Proceedings of the 13th ACM SIGOPS Asia-Pacific Workshop on Systems, 2022, pp. 18–28.

[5] X. Wei, R. Cheng, Y. Yang, R. Chen, H. Chen, Characterizing off-path {SmartNIC} for accelerating distributed systems, in: 17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23), 2023, pp. 987–1004.
[6] Smart SFP product page, https://www.smartsfp.com/, 2023.
[7] Microchip datasheet of an SFP+ module with an integrated FPGA, https://ww1.microchip.com/downloads/aemDocuments/documents/FPGA/ApplicationNotes/ApplicationNotes/PolarFire_FPGA_SFP_Plus_Module_AN4364_VA.pdf, 2023.
[8] Accedian intelligent SFP with integrated FPGA for monitoring applications, https://accedian.com/wp-content/uploads/2023/01/Accedian_Datasheet_Skylight-sensor-SFP-compute_JAN2023.pdf, 2023.
[9] C. Smiliotopoulos, G. Kambourakis, C. Kolias, Detecting lateral movement: A systematic survey, Heliyon (2024).
[10] A. Greco, G. Pecoraro, A. Caponi, G. Bianchi, Advanced widespread behavioral probes against lateral movements, Int. J. Inf. Secur. Res 6 (2016) 651–659.
[11] A. Niakanlahiji, J. Wei, M. R. Alam, Q. Wang, B.-T. Chu, {ShadowMove}: A stealthy lateral movement strategy, in: 29th USENIX Security Symposium (USENIX Security 20), 2020, pp. 559–576.
[12] N. Hu, M. Ye, S. Wei, Surviving information leakage hardware trojan attacks using hardware isolation, IEEE Transactions on Emerging Topics in Computing (2019).
[13] A. M. Azab, P. Ning, X. Zhang, Sice: a hardware-level strongly isolated computing environment for x86 multi-core platforms, in: Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS '11, Association for Computing Machinery, New York, NY, USA, 2011, p. 375–388. URL: https://doi.org/10.1145/2046707.2046752. doi:10.1145/2046707.2046752.
[14] ARM Trustzone, https://www.arm.com/technologies/trustzone-for-cortex-a, 2023.
[15] Intel Software Guard Extensions, https://www.intel.com/content/www/us/en/developer/tools/software-guard-extensions/overview.html, 2023.
[16] A. Rivitti, R. Bifulco, A. Tulumello, M. Bonola, S. Pontarelli, ehdl: Turning ebpf/xdp programs into hardware designs for the nic, in: Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3, ASPLOS 2023, Association for Computing Machinery, New York, NY, USA, 2023, p. 208–223. URL: https://doi.org/10.1145/3582016.3582035. doi:10.1145/3582016.3582035.
[17] M. Eskandari, Z. H. Janjua, M. Vecchio, F. Antonelli, Passban ids: An intelligent anomaly-based intrusion detection system for iot edge devices, IEEE Internet of Things Journal 7 (2020) 6882–6897. doi:10.1109/JIOT.2020.2970501.
[18] M. Bachl, J. Fabini, T. Zseby, A flow-based IDS using machine learning in ebpf, https://arxiv.org/abs/2102.09980, 2021.
[19] L. Deri, S. Sabella, S. Mainardi, P. Degano, R. Zunino, Combining system visibility and security using ebpf., in: ITASEC '19, 2019.
[20] Suricata, Suricata IDS Website, 2023. https://suricata.io/.
[21] G. Bertin, Xdp in practice: integrating xdp into our ddos mitigation pipeline, in: Technical Conference on Linux Networking, Netdev, volume 2, The NetDev Society, 2017, pp. 1–5.
[22] S. Miano, M. Bertrone, F. Risso, M. V. Bernal, Y. Lu, J. Pi, Securing linux with a faster and

scalable iptables, https://doi.org/10.1145/3371927.3371929, 2019.
[23] Isovalent, Cilium, https://cilium.io/, 2023.
[24] Tigera, Project calico, https://www.tigera.io/project-calico/, 2023.
[25] G. Budigiri, C. Baumann, J. T. Mühlberg, E. Truyen, W. Joosen, Network policies in kubernetes: Performance evaluation and security analysis, in: 2021 Joint European Conference on Networks and Communications & 6G Summit (EuCNC/6G Summit), IEEE, 2021, pp. 407–412.
[26] Microchip, Polarfire soc, https://www.microchip.com/en-us/products/fpgas-and-plds/system-on-chip-fpgas/polarfire-soc-fpgas, 2023.
[27] Intel, Intel agilex soc, https://www.intel.com/content/www/us/en/products/details/fpga/agilex/9/direct-rf-series.html, 2023.
[28] MeherRushi, Flowsentryx - an xdp-based dos and ddos mitigation framework, https://github.com/MeherRushi/FlowSentryX, 2023.