Rethinking NIDS Rule-Based Pre-Filtering

Henrique B. Brum*†, Luis A. D. Knob†, Tiago Ferreto‡, Domenico Siracusa*
*University of Trento
henrique.beckerbrum@unitn.it, domenico.siracusa@unitn.it
†Fondazione Bruno Kessler
l.diasknob@fbk.eu
‡Pontifical Catholic University of Rio Grande do Sul
tiago.ferreto@pucrs.br

***Abstract*—Surging network traffic is pushing signature-based Network Intrusion Detection Systems (NIDSs) to their limits, as they struggle to inspect every incoming packet. The high computational cost of analyzing each packet and matching it against large rulesets can lead to system saturation and missed attacks. Rule-based pre-filtering can reduce this cost by forwarding only potentially malicious packets to the NIDS. However, existing work has largely overlooked the impact of pre-filtering on the NIDS's attack detection performance. Our analysis shows that current methods disrupt attack detection because they discard packets that do not match the pre-filtering rules, ignoring the fact that NIDSs require additional flow information (e.g., the TCP handshake) beyond the malicious packets to process flows and detect attacks properly. To address this, we propose *eRBF*, a new rule-based pre-filtering approach that pre-filters traffic not only according to the rules but also based on the NIDSs' flow processing requirements. Experimental results with Snort demonstrate that *eRBF* achieves the desired balance, forwarding less than 22% of all packets across two well-known datasets while maintaining the attack detection above 95%.**

***Index Terms*—NIDS, Security, Pre-filtering, Performance**

I. Introduction

Network Intrusion Detection Systems (NIDSs) play a critical role in defending networks against malicious activity. Although AI-based NIDS are on the rise, traditional signature-based NIDSs are still widely used, with Snort [1] being one of the most prominent examples [2]. These systems operate by comparing packets to rules, where each rule contains signatures (characteristics) of known attacks. Due to the ever-increasing network traffic and number of attacks, the packet-rule comparison stage can cause signature-based NIDS to saturate, leading to dropped packets and missed attacks. In [3], the authors have shown that even with fast packet acquisition mechanisms, Snort could process only up to 60 Gbps before consuming all the available CPU and dropping packets.

Common approaches to addressing NIDS saturation include improving internal algorithms [4], bypassing the traditional network stack [5], or designing high-speed NIDSs from scratch [6]. In parallel to these methods, researchers have explored reducing the number of packet–rule comparisons by pre-filtering traffic [7]–[14]. The most common pre-filtering method is flow (or packet) sampling, where only a subset of packets from each flow is forwarded to the NIDS. While used in commercial standards such as sFlow[1] and studied in academia [7], [8], flow sampling faces a key challenge: selecting an effective sampling threshold for diverse traffic patterns. Forwarding too many packets reduces filtering benefits, while forwarding too few may cause attacks to go undetected, making it difficult to strike the right balance with static thresholds.

Several studies [9]–[14] have leveraged NIDS rules to pre-filter network traffic and achieve this balance. The idea is to offload a simplified version of the rules into the data plane (i.e., FPGAs, P4 switches, or eBPF) and forward only a subset of the traffic that is potentially malicious. However, most works have focused primarily on implementation details and the volume of packets pre-filtered, while overlooking the impact of pre-filtering on the NIDS's attack detection performance. By either not evaluating this aspect at all [9], [10] or providing little explanation for poor detection results [12], [13], current works have neglected this aspect, leaving the applicability of rule-based pre-filtering uncertain.

Our initial experiments with current rule-based methods—now evaluating the attack detection performance—showed that filtering solely by rules disrupts NIDS detection capabilities. This occurs because NIDSs often require additional packets from a flow (e.g., the TCP handshake), beyond the malicious ones, to identify attacks correctly. To address this, we propose ***eRBF*** (extended Rule-Based pre-Filtering), a rule-based pre-filter that reduces traffic forwarded to the NIDS while preserving attack detection performance. Like existing work, *eRBF* uses simplified NIDS rules to identify and forward potentially malicious packets. However, unlike existing work, it also forwards additional packets that may not match any rules, but are essential for correct flow processing and attack detection. We evaluated *eRBF* against flow sampling and existing rule-based methods using Snort and two NIDS datasets. The results show that *eRBF* achieves the best balance between reducing the number of packets processed by the NIDS and preserving the attack detection performance, demonstrating its effectiveness in tackling NIDS saturation.

II. Background and Related work

A. Signature-Based NIDS

Network attacks typically follow specific patterns: they target known vulnerabilities, execute certain commands, or generate traffic at a high packet rate in the general case

[1]https://sflow.org/

978-3-903176-75-1 ©2025 IFIP

of DoS attacks. As a result, once an unknown attack has been discovered and analyzed, it is often straightforward to create a rule that identifies the associated malicious behavior. Snort, one of the most widely used signature-based NIDSs [2], was explicitly designed for this purpose. It enables network operators to craft custom rules or use existing ones to protect networks against well-known attacks.

Rules consist of two parts: the *header*, which specifies the action to take upon a match, the network protocol, source and destination IPs, and ports; and the *options*, which define additional characteristics to inspect in the network traffic. The *options* section allows operators to describe packet characteristics that indicate an attack in detail. These include fields such as *content* and *PCRE* (Perl-Compatible Regular Expression), which inspect a packet's payload for specific byte patterns or strings. Other fields can define rate-based conditions to detect attacks like DoS. By crafting effective rules and keeping rulesets up to date, NIDS can safeguard networks against a wide range of threats.

B. Pre-Filtering for Signature-Based NIDS

As traffic volume and the variety of network attacks grow, the workload on signature-based NIDSs increases significantly. This becomes especially problematic in high-speed networks, where processing all packets at line rate becomes unmanageable, causing NIDS saturation and undetected attacks. By pre-filtering packets and forwarding only a small subset of traffic to the NIDS, it is possible to mitigate NIDS saturation. The traditional method for pre-filtering is flow (or packet) sampling, where a switch mirrors a fraction of the incoming packets to the NIDS. Although there are various sampling strategies, most of them are based on static thresholds that define the sampling frequency. For example, sFlow uses a basic sampling technique that forwards one out of every N packets[2]. Similarly, sampling strategies proposed in academic NIDS research [7], [8] depend on static thresholds to define how many packets are filtered.

Although flow sampling is easy to deploy, setting thresholds that effectively reduce the network traffic while maintaining accurate attack detection for different traffic types and NIDSs is challenging. To overcome this, researchers have proposed rule-based pre-filtering, where a simplified NIDS ruleset is offloaded to a pre-filtering device that discards unnecessary mirrored packets before they reach the NIDS. This filtering can be implemented inside the mirroring switch using P4 [12], [13], on intermediate devices such as FPGAs [9]–[11], or even on the NIDS server itself via eBPF [14].

Despite this significant body of work, a common limitation remains: the lack of attention to attack detection performance after pre-filtering. Some studies provide no evaluation of this metric [9], [10], while others report noticeable drops in detection accuracy without analyzing the root causes [12], [13]. In some cases, the evaluation does not include open-source NIDSs [14], thereby reducing their applicability, or only briefly suggests possible causes for performance drops without providing a comprehensive solution [11]. By neglecting this essential aspect, existing methods fail to deliver solutions that are viable for real-world deployment.

[2]https://sflow.org/packetSamplingBasics/

III. Extended Rule-Based Pre-Filtering for NIDSs

Rule-based pre-filtering can reduce the number of packets a NIDS must process and help prevent saturation. The main challenge is balancing three goals: (1) an implementation that operates at line rate without becoming the bottleneck, (2) reducing the number of packets reaching the NIDS, and (3) maintaining the expected attack detection performance after pre-filtering. While the first two goals have been well studied, the third goal—preserving detection performance—has received comparatively little attention, particularly when integrating with open-source NIDSs like Snort.

Our initial analysis using existing rule-based methods and Snort revealed that, although they can reduce traffic load, these methods also compromise attack detection. To understand why, we analyzed Snort's source code and documentation and found that it tracks flows to reassemble data and correctly process transport and application protocols. This means Snort requires additional packets, such as those in the TCP handshake, alongside the malicious ones, to identify attacks. Current pre-filtering approaches forward only rule-matching packets (i.e., potentially malicious ones), missing these essential flow-related packets and causing the observed decline in attack detection performance.

Based on this investigation, we reconsidered the priorities in designing a rule-based pre-filter and developed *eRBF*. Rather than focusing on low-level implementation details tied to specific technologies (goal 1), *eRBF* is a rule-based pre-filter that focuses on reducing the traffic reaching the NIDS (goal 2) while preserving attack detection performance (goal 3). To this end, we identified several additional packets that, although they may not directly match any rule, are nonetheless essential for the NIDSs to detect malicious traffic accurately:

- **TCP Connection.** Since TCP is a connection-oriented protocol, Snort requires a sequence of packets to interpret a flow accurately. Although Snort does not need every packet in a TCP flow, those related to connection establishment and termination (i.e., SYN, SYN+ACK, RST, and FIN) are essential and therefore forwarded by default. Additionally, the first ACK after a suspicious packet flagged by *eRBF* is also forwarded, as it provides crucial information for Snort's TCP inspector.
- **FTP Request and Response.** For FTP traffic, Snort might not be able to properly parse a suspicious FTP request or response without the previous request or subsequent response. Accordingly, both FTP request and response packets are forwarded by default.
- **DNS Query**. Lastly, we also observed unreported attacks for DNS response packets, even when the DNS response packets containing malicious content were forwarded to Snort. Further analysis showed that the corresponding DNS queries were essential for Snort to evaluate those

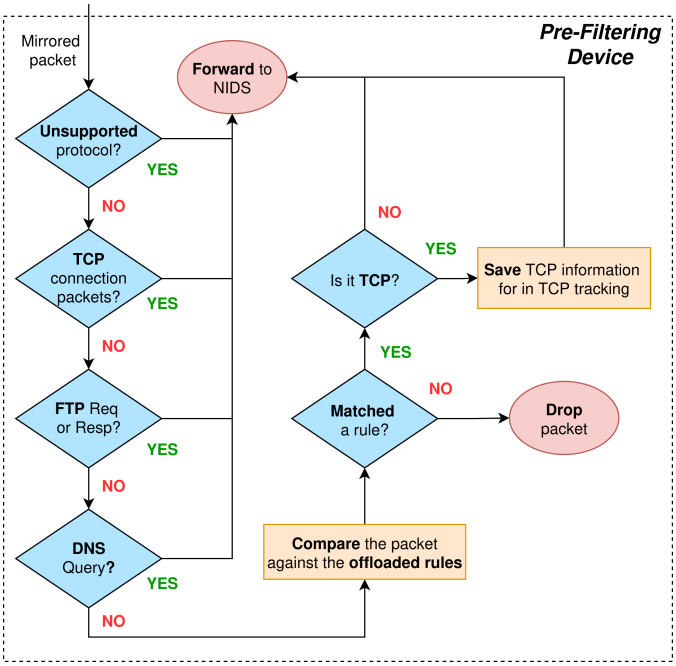

Fig. 1: Flow of a mirrored packet within *eRBF*.

DNS responses as malicious properly. As a result, DNS queries are forwarded by default.

Fig. 1 illustrates the pre-filtering process for each mirrored packet inside *eRBF*. Packets using unsupported protocols are forwarded directly to the NIDS, since the pre-filter currently handles only common protocols such as TCP, UDP, and HTTP. This set of supported protocols can be extended according to the main protocols in the network. Packets with supported protocols are first checked against the predefined conditions mentioned before and then against the offloaded ruleset. If no match is found in both comparisons, the packet is dropped; otherwise, it is forwarded to the NIDS for final inspection. For TCP packets that trigger a rule, the pre-filter also records the flow's five-tuple to ensure subsequent ACKs are forwarded.

A final modification over existing work, motivated by the additional packets forwarded by default in our design, was to improve the pre-filtering and make it more fine-grained. Current work typically offloads highly simplified rules, limited to header fields (e.g., Layer 4 ports and IP TTL) [12], [13] or a single payload *content* value [14]. In contrast, *eRBF* incorporates all *content* fields of a rule as well as *PCRE* fields, allowing for more precise pre-filtering. This reduces the number of packets reaching the NIDS and addresses a common issue of overly broad rule offloading. Oversimplified rules often become too generic, forcing existing methods to discard them and risk missing relevant packets. By preserving most rule fields, *eRBF* avoids forwarding excessive irrelevant traffic, thus improving the overall pre-filtering effectiveness.

IV. Evaluation Setup

To evaluate *eRBF* without being constrained by technical details of specific tools, we developed a Python-based pre-filtering simulator. It reads a PCAP file, filters the packets according to the selected strategy, and saves the remaining packets to another PCAP file, which the NIDS then analyzes. The simulator's logic depends on the pre-filtering method. For flow sampling, a hash map is used to track flows and apply thresholds to determine which packets are included in the final PCAP. For rule-based pre-filtering, rules are parsed, simplified, and grouped by protocol, IP, and port to reduce comparisons. During simulation, packets are compared sequentially against the relevant rules, and, in the case of *eRBF*, they also perform the additional predefined conditions. The simulator's code and the evaluation materials can be found in our repository[3].

We compared *eRBF* against the following strategies:

- *Flow Sampling*: Inspired by [8], the first N packets of each arriving flow are forwarded to the NIDS, and N is reset after T seconds. We experimented with values of 5, 10, 25, and 50 for both N and T.
- *Header-Only Rule-Based Pre-Filtering:* The simplest rule-based approach, based on [12], [13], offloads only fields directly available in the packet header (e.g., IP addresses, TCP flags) or easily derived from them (e.g., TCP payload size). To improve pre-filtering, we applied a strategy from [12] that excludes rules using the *any* keyword in more than two of the following fields: source IP, destination IP, source port, and destination port.
- *Fast-Pattern Rule-Based Pre-Filtering:* Based on [14], this method extends the header-only approach by offloading the *fast-pattern* content. As noted in [14], the *fast-pattern* check is the first payload inspection step in NIDSs and is designed to include the most discriminative strings or byte sequences for efficient filtering.

Lastly, we used *Snort 3.7.4* and the Snort 3 Registered[4] ruleset with 46 346 rules enabled. As network input, we used the CICIDS2017 [15] and CICIoT2023 [16] datasets, which together cover a broad range of traffic patterns and network attacks.

V. Results

The main focus of this paper is the balance between packets forwarded to the NIDS and preserving the attack detection performance. This balance reflects how effectively a pre-filtering method can address the NIDS saturation issue without compromising the NIDS's ability to detect malicious activity.

Alerts are used to evaluate attack detection performance because they are the default action when a rule matches a packet. Each alert is defined as a unique combination of a flow's five-tuple and a rule ID, so repeated alerts from the same rule on the same flow are counted only once. It is important to note that not all alerts correspond directly to labeled attacks in the datasets: rules can generate alerts for benign or unlabeled flows, and some labeled attacks may go undetected if the ruleset lacks corresponding rules. This does not affect our evaluation, as the goal is not to assess the completeness of

[3]https://github.com/daisyfbk/Pre-Filtering-Simulator
[4]https://www.snort.org/downloads/registered/

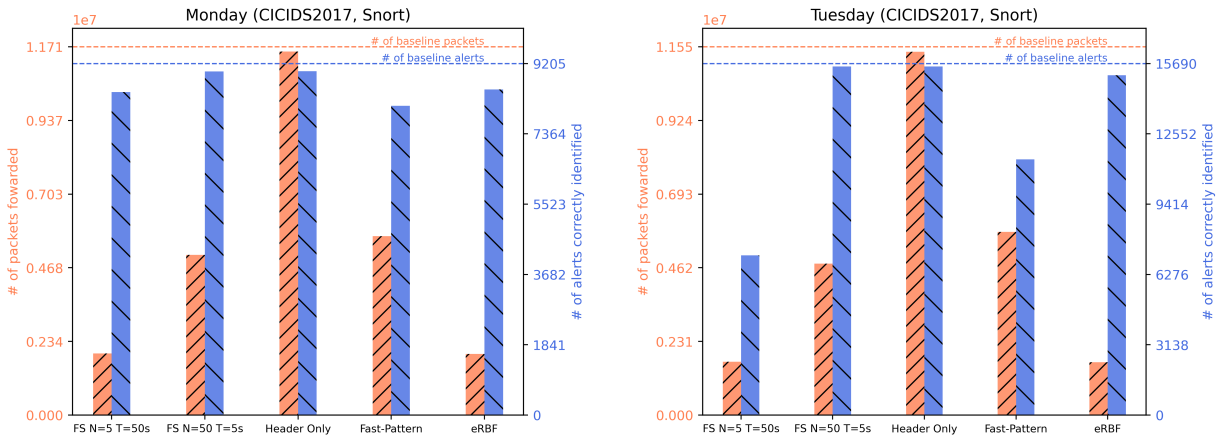

Fig. 2: Packets forwarded vs alerts correctly identified for **CICIDS2017**.

the ruleset or the quality of the dataset, but to measure how different pre-filtering strategies perform under realistic traffic conditions.

The results demonstrating this balance are first presented separately for each dataset, followed by an overview of all traces. The two selected graphs in the dataset-specific sections either illustrate trends commonly observed across experiments or highlight atypical behaviors. We display two configurations for the flow sampling (FS in the graphs) approach: one that achieves the highest filtering rate and another that yields the highest number of correctly identified alerts. Finally, we established a common baseline for all the pre-filtering strategies, corresponding to Snort processing the full, unfiltered trace, which represents the ideal deployment scenario.

A. *CICIDS2017*

Fig. 2 displays the results for two of the five traces for the CICIDS2017 dataset using Snort. The graphs compare the number of packets forwarded in absolute terms, on the left, with the number of alerts correctly identified, also in absolute terms, on the right. In both graphs, the number of packets and the number of alerts when Snort processes all the packets for a trace (baseline) are indicated by the lines on top of them.

These graphs illustrate the core challenge of flow sampling: balancing how much is filtered without decreasing the NIDS attack detection capabilities for different traffic types. While rule-based pre-filtering is not a guaranteed fix—as seen in the results for the *Header Only* and *Fast-Pattern* approaches—it can achieve a better balance when the NIDS's flow processing logic is taken into account, as demonstrated by *eRBF*. By leveraging more rule information and forwarding additional flow-relevant packets, *eRBF* filters more traffic while maintaining a higher alert correlation.

B. *CICIoT2023*

eRBF's results for the CICIoT2023 dataset showed a similar trend to those observed with CICIDS2017. However, some effects were more pronounced because the traces contain fewer packets and exhibit more homogeneous traffic patterns, as shown in Fig. 3. For example, the *Backdoor_Malware* trace demonstrates a perfect performance by *eRBF*: all alerts were correctly identified, while only about 15% of the network traf-

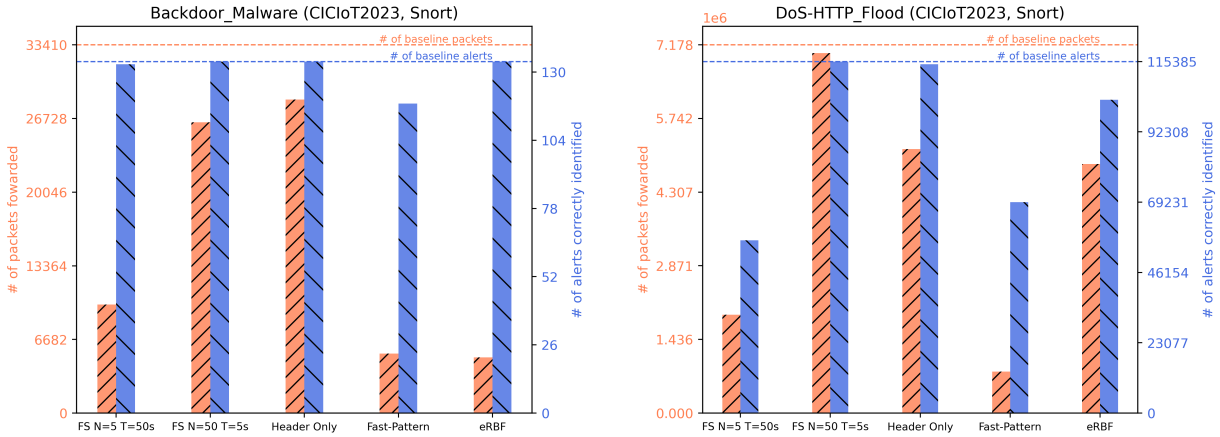

Fig. 3: Packets forwarded vs alerts correctly identified for **CICIoT2023**.

fic was forwarded. In contrast, the *DDoS_HTTP* trace posed a challenge regarding packet filtering. This trace includes many small TCP flows, resulting in *eRBF* forwarding by default a large volume of SYN, SYN+ACK, FIN and RST packets, therefore reducing the overall filtering effectiveness.

C. Overview

Although individual traces are useful for highlighting specific behaviors, Fig. 4 provides an overview of the datasets as a whole. In this figure, the packets forwarded and alerts correctly identified by each strategy were first calculated as percentages of the baseline for each trace. These percentages were then averaged across both datasets and presented in a scatter plot.

We can generalize trends observed in earlier images from this overview graph. Flow sampling can achieve high alert correlation or significantly reduce traffic, but balancing both objectives for diverse traffic patterns is challenging. Rule-based pre-filtering methods that ignore a NIDS's flow processing logic (triangle and square markers) also struggle to achieve this balance. For instance, filtering based only on header-related fields, even when removing broad rules, fails to pre-filter packets for the NIDS. Including the *fast-pattern* content can sometimes improve filtering but degrades the alert detection, as necessary, but non-suspicious, packets have a higher chance of being excluded.

eRBF achieves the best balance among the evaluated methods. For CICIDS2017, it forwarded only 19% of the traffic while maintaining a 95% alert correlation. For CICIoT2023, it forwarded 21% of packets, slightly higher, but the attack detection improved to 96%. These results support our hypothesis: by going beyond simple rule matching and considering the NIDS's flow-processing behavior, rule-based pre-filtering can mitigate NIDS saturation without compromising detection performance.

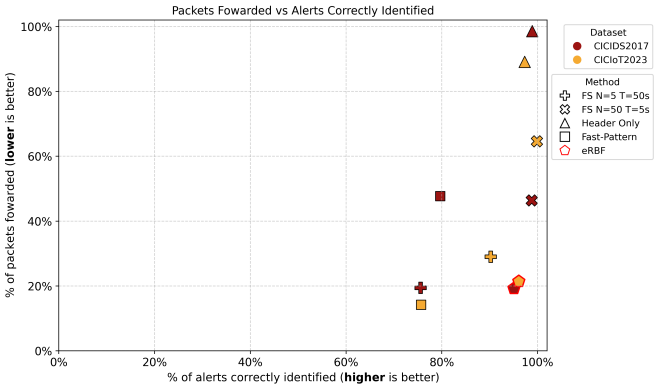

Fig. 4: Overview of packets forwarded vs alerts correctly identified.

VI. Conclusion

The ever-increasing volume of network traffic and cyber attacks can cause signature-based NIDSs to saturate, leading to missed attacks. Rule-based pre-filtering provides a promising approach to reducing the traffic these systems must process. However, existing methods have struggled to minimize network traffic without compromising the NIDS's ability to detect attacks. To address this limitation, we introduced *eRBF*, a rule-based pre-filter that significantly reduces the traffic reaching the NIDS while preserving its attack detection performance. Our experiments with two well-known datasets demonstrate that *eRBF* achieves the best balance between traffic reduction and detection accuracy, forwarding less than 22% of all packets while preserving Snort's attack detection above 95%. Future work will focus on expanding *eRBF* to provide pre-filtering for Suricata and designing an efficient hardware implementation.

References

[1] Cisco, "Snort: Network intrusion detection & prevention system," https://www.snort.org/, accessed on: (05/2025).

[2] A. Waleed, A. F. Jamali, and A. Masood, "Which open-source ids? snort, suricata or zeek," *Computer Networks*, vol. 213, p. 109116, 2022.

[3] Q. Hu, S.-Y. Yu, and M. R. Asghar, "Analysing performance issues of open-source intrusion detection systems in high-speed networks," *Journal of Information Security and Applications*, vol. 51, p. 102426, 2020.

[4] X. Wang, O. O'Halloran, S. Ravisundar, W. Andralojc, D. Doherty, H. Zhu, and M. Koyama, "Accelerate snort performance with hyperscan and intel xeon processors on public clouds," Intel Corporation, Tech. Rep., 2023.

[5] M. Sheeraz, M. H. Durad, S. Tahir, H. Tahir, S. Saeed, and A. M. Almuhaideb, "Advancing snort ips to achieve line rate traffic processing for effective network security monitoring," *IEEE Access*, vol. 12, pp. 61 848–61 859, 2024.

[6] Z. Zhao, H. Sadok, N. Atre, J. C. Hoe, V. Sekar, and J. Sherry, "Achieving 100gbps intrusion prevention on a single server," in *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, 2020, pp. 1083–1100.

[7] J. Kučera, L. Kekely, A. Piecek, and J. Kořenek, "General ids acceleration for high-speed networks," in *2018 IEEE 36th International Conference on Computer Design (ICCD)*. IEEE, 2018, pp. 366–373.

[8] B. Lewis, M. Broadbent, C. Rotsos, and N. Race, "4midable: Flexible network offloading for security vnfs," *Journal of Network and Systems Management*, vol. 31, no. 3, p. 52, 2023.

[9] M. Attig and J. Lockwood, "Sift: Snort intrusion filter for tcp," in *13th Symposium on High Performance Interconnects (HOTI'05)*. IEEE, 2005, pp. 121–127.

[10] H. Song, T. Sproull, M. Attig, and J. Lockwood, "Snort offloader: A reconfigurable hardware nids filter," in *International Conference on Field Programmable Logic and Applications, 2005.* IEEE, 2005, pp. 493–498.

[11] S. Teofili, E. Nobile, S. Pontarelli, and G. Bianchi, "Ids rules adaptation for packets pre-filtering in gbps line rates," *Trustworthy Internet*, pp. 303–316, 2011.

[12] B. Lewis, M. Broadbent, and N. Race, "P4id: P4 enhanced intrusion detection," in *2019 IEEE conference on network function virtualization and software defined networks (NFV-SDN)*. IEEE, 2019, pp. 1–4.

[13] K. Tavares and T. C. Ferreto, "P4-onids: A p4-based nids optimized for constrained programmable data planes in sdn," *Anais do XXXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, 2021, Brasil.*, 2021.

[14] S.-Y. Wang and J.-C. Chang, "Design and implementation of an intrusion detection system by using extended bpf in the linux kernel," *Journal of Network and Computer Applications*, vol. 198, p. 103283, 2022.

[15] I. Sharafaldin, A. H. Lashkari, A. A. Ghorbani *et al.*, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." *ICISSp*, vol. 1, no. 2018, pp. 108–116, 2018.

[16] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, "Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment," *Sensors*, vol. 23, no. 13, p. 5941, 2023.