High Performance Network Metadata Extraction Using P4 for ML-based Intrusion Detection Systems

Nicholas Gray, Katharina Dietz, Michael Seufert, Tobias Hossfeld
University of Würzburg
Würzburg, Germany
{nicholas.gray, katharina.dietz, michael.seufert, tobias.hossfeld}@uni-wuerzburg.de

***Abstract*—Today's communication networks process an increasing amount of traffic, while simultaneously providing services to a larger and more diverse quantity of devices. This enhances the complexity of the network and imposes a larger attack space, impacting network management and security efforts. Deployed hardware middle-boxes, likefirewalls and Intrusion Detection Systems (IDSs) often lack theflexibility to adapt to this dynamic environment, which Network Function Virtualization (NFV) addresses by implementing these services in software. Yet, this may impose a bottleneck, due to the absence of hardware acceleration. To mitigate this drawback, the functionality can be offloaded to programmable hardware, using P4. In this work we implement an IDS, capable of operating in core and backbone networks up to$100Gbps$. This is achieved by using the hardware acceleration of P4-enabled Intel© Tofino™ switches for high performance metadata extraction, in order to train an ML-based detection engine. The system is evaluated regarding its throughput and obtainable aggregation levels as well as its accuracy for detecting a variety of network attacks.**

***Index Terms*—P4, Machine Learning, Intrusion Detection, High Performance Networking, Data Plane Development Kit (DPDK), Network Function Virtualization (NFV)**

I. Introduction

Driven by the expanding impact of recent trends like IoT, Industry 4.0 and 5G, modern communication networks are subject to an everlasting increase of network traffic and number of connected devices as well as a larger variety of supported applications [1]. Therefore, the overall complexity is augmented and network management and security have to be constantly adapted, as the sophistication and amount of cyber attacks against these systems is on the rise [2].

To cope with the dynamics of these threats, proprietary hardware-based defenses, i.e.,firewalls, are transformed to software-based solutions to provide a higher degree offlexibility and to ease their deployments within cloud environments. Furthermore, additional defensive barriers are established and aggregated to form Security Information and Event Management (SIEM) systems to assist in the process of anomaly and zero day detection. Advances in thefields of Artificial Intelligence (AI) and Machine Learning (ML) have further evolved these system by enhancing the accuracy and their ability of identifying prior unknown attacks.

Yet, in contrast to their hardware-based counterparts, which rely on Application-Specific Integrated Circuits (ASIC) for fast packet processing, software-based solutions may impose a bottleneck [3] on to the network. This is especially the case for high bandwidth core and backbone networks, as every single packet needs to be inspected to provide an adequate degree of protection. To tackle this challenge, a possible solution is to incorporate Software-defined Networking (SDN) to offload the functionality to programmable networking devices. In this context, the network programming language P4 and its supporting platforms like the Intel© Tofino™ are gaining an increased popularity. An experimental approach called In-band Network Telemetry (INT) [4], [5] already incorporates P4 for network monitoring purposes, by using the programmable switch for metadata extraction and aggregation.

This raises the question if P4 can be as well applied to security related systems, hence providing the scalability needed for an ML-based Intrusion Detection System (IDS), which is deployable in high bandwidth networks.

The contribution of this work is $a)$ the design and implementation of an IDS using programmable switches as hardware accelerators, $b)$ an in-depth evaluation regarding the achievable throughput and metadata aggregation of each component in high bandwidth networks up to $100Gbps$ and $c)$ a comparison of the derived ML detection model to the IDS Suricata [6].

This work is structured as follows. Section II and III give an introduction to the relevant background and establish an overview of the related work, respectively. In Section IV the system design and technical limitations of the P4-enabled hardware are detailed. After specifying the testbed and measurement parameters in Section V, the evaluation results of the system are given in Section VI. A summary and outlook to future work is described in Section VII.

II. Background

In this section we provide the required background to the concepts and technologies used within this work.

Traditionally, telecommunication service providers (TSP) deployed specialized and proprietary middleboxes within the data path of their networks to implement load balancers or firewalls [7]. This resulted in increasing capital expenditures (CAPEX) and operational expenditures (OPEX), as higher traffic demands had to be met.

Network Function Virtualization (NFV) tries to mitigate these negative effects by deploying network functions as software-

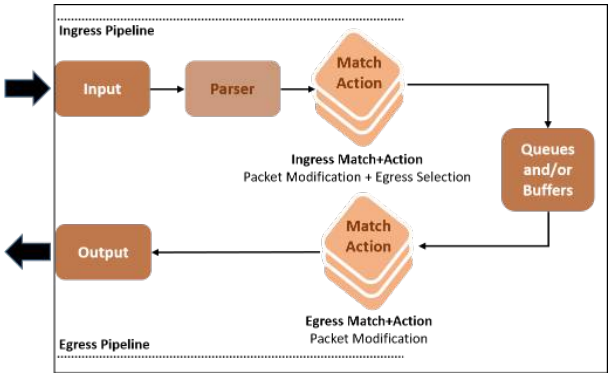

Fig. 1: Simplified P4 abstract forwarding model.

based solutions on Commodity-Off-the-Shelf (COTS) hardware, hence providing a higherflexibility by accelerated development cycles and decreased costs.

Yet, relying on COTS servers, NFV induces a high overhead on per packet operations, due to the lack of specialization. As this results in performance impairments, more sophisticated packet processing technologies have emerged to address these issues [8]. For example, the Linux kernel has been optimized in regards to its interrupt moderation capabilities and frameworks like the Data Plane Development Kit (DPDK) bypass the networking stack of the operating system to provide direct access to the network interface card (NIC).

Another solution is to offload the functionality to programmable switching hardware. In this context, the programming language Programming Protocol-Independent Packet Processor (P4) [9] has gained increased interest within the research community. It provides domain specific programming constructs, which are optimized and streamline the development. As P4 aims to be deployed on programmable switching chips it specifies the processing architecture as depicted in Figure 1.

Here, each incoming packet is subject to a processing pipeline. In thefirst step the packet is parsed according to the header definitions and the order is specified by a parse graph. As these definitions are adjustable by the programmer, P4 enables the development of new protocols from scratch beginning with the data link layer. Then, the extracted information is matched against a set of tables, which define and apply specific actions to the packet, e.g., port selection or modifications. Whereas the actions and matchfields need to be specified at compile time, their entries can be altered and configured at run time via an interface to the control plane. Subsequently the packet is transferred to the buffer and queuing module. Before the packet is emitted through the selected port, it is once again subject to a set of egress*match & action tables*, which are similar to those of the ingress.

The Intel© Tofino™ chip is based on the Protocol-Independent Switch Architecture (PISA) [10] and is one compliant implementation of the aforementioned architecture. PISA augments the reference architecture by implementing a packet replication engine and traffic manager within the buffer and queuing module as well as providing a parser before and a deparser after the ingress and egress*match & action tables*. Whereas thefirst augmentation enables the ability to clone and recirculate packets, the second adaptation allows for resource sharing. In addition, the architecture provides access to Static Random-Access Memory (SRAM) and Ternary Content Addressable Memory (TCAM) as well as certain arithmetic operations within its Match-Action Units (MAUs).

As the sophistication of attacks against computer networks is rising, a singlefirewall is no longer a sufficient protection. Hence, additional systems such as Intrusion Detection Systems (IDS) are deployed. Sharing the common goal of detecting and logging a myriad of attacks, different implementations and categories of these systems exist [11], [12]. As the IDS is required to inspect the communication it can either be deployed on every host (HIDS) or in a central location within the network (NIDS), which has the advantage of being able to correlate attacks across multiple devices. An IDS can be further classified by the employed detection mechanism, e.g., signature or anomaly based. In contrast to signature based detection mechanisms, anomaly based solutions are able to identify previously unseen attacks. This category has largely benefited from the recent advances in thefields of Artificial Intelligence (AI) and Machine Learning (ML). Especially, supervised learning algorithms, i.e., decision-tree based algorithms have proven to provide viable classifier models. At last, frameworks like Scikit-learn [13] make these algorithms available to a broad community of developers.

In this work, we leverage these recent developments and investigate the applicability of softwarized network functions for NIDSs in today's high speed networks. For this, we implement a NIDS, which relies on a programmable P4 switch for feature extraction, on DPDK for feature collection and Scikit-learn for ML-based anomaly detection, and evaluate its performance in a 100Gbps testbed.

III. Related Work

In this section we detail the related work and describe similarities and differences to our approach.

Several performance evaluations of efficient packet processing technologies have been conducted, as they are key elements affecting the performance of an IDS [11]. In [14] the signature-based IDSes Suricata and Snort are evaluated, showing that their performance decreases with increasing load as these systems rely heavily on the available CPU resources. In [15] an anomaly-based IDS is proposed, which uses pre-aggregated Netflow data in order to handle traffic up to $1Gbps$. A similar approach is discussed in [16] which evaluates different machine learning algorithms against 18 features. The authors conclude that decision-tree algorithms produce among the best results and that the applied packet processing technology is the limiting performance factor. In this work, we compare the detection accuracy of our anomaly-based IDS against the state-of-the-art IDS Suricata. In contrast to the presented anomaly-based IDSes, we evaluate our system with speeds up to $100Gbps$ and use P4-enabled switches to extract and pre-aggregate 31 flow-based features.

The impact of specialized programmable hardware to increase the packet processing performance is investigated in [17] and in [18] using Field Programmable Gate Arrays (FPGAs) and Graphics Processing Units (GPUs), respectively. Although enhancing the performance of NFV solutions, bottlenecks may occur while transferring the data to and from the NIC. To mitigate this effect FPGA-based SmartNICs provide a direct network connector, which is wired to the processing units. The authors of [19] propose a framework which enables the shared usage of a SmartNIC across several NFV solutions with speeds up to $200Mpps$. Focusing rather on the accessibility than the performance gains of SmartNICs, [20] ports eXpress Data Path (XDP) programs to the platform. Another approach to decrease the steep learning curve of FPGA-based SmartNICS is taken by [21], in which a P4 compiler is introduced for the NetFPGA SUME platform. For the IDS solution presented in this work, we incorporate the the Intel© Tofino™ platform as packet processing accelerator, since it provides direct P4 support and is able to process$100Gbps$.

The use of P4-enabled switches for monitoring purposes is demonstrated by the In-band Network Telementry (INT) framework [5]. The main goal of the framework is to extract metadata from the switch, e.g., queue utilization orflow counters. The extracted information is then added as header to the packet and may be processed by an external instance. Following this approach, [22] and [23] implement a heavy hitter detection, which is able to operate at line rates up to $100Gpbs$. In a security related context, mechanisms for detecting Distributed Denial of Service (DDoS) attacks by employing P4-enabled switches are presented in [24], [25]. A more complete approach is taken by P4ID [26], which compiles stateless intrusion detection rules directly to a P4-enabled switch, while redirecting a certain amount of packets perflow for stateful inspection to an IDS, hence reducing the overall load on this system. The aspect of aggregatingflow-based data is shown in [27], which utilizes hash collisions during the recording of statistics as trigger to export the collected metadata. In contrast to INT, we emit the extracted data out-of-band via a dedicated port due to security considerations. Yet, we employ a similar metadata extraction technique, which is combined with the aforementioned aggregation mechanisms. At last we aim for implementing a full IDS, capable of detecting a wider variety of attacks instead of focusing solely on DDoS. Whereas the goal of reducing the load on the decision making instance of the IDS is shared between P4ID and this work, we focus on reducing it by simply forwarding the relevant and aggregated features instead of the full packets.

IV. System Design & Limitations

In the following, we detail the system design of the proposed IDS solution and provide insights into the details of the implementation. Furthermore, we discuss the applied solutions to limitations imposed by the targeted P4 platform.

System Design. The goal of the proposed NIDS is to provide a high detection accuracy for a broad range of network attacks by inspecting each packet in high bandwidth networks. An overview of the system components and their interconnections is depicted in Figure 2.

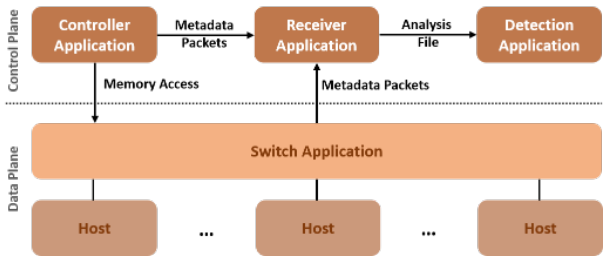

Fig. 2: Overview of the proposed NIDS.

The system consists of four main modules, e.g., the Controller, Receiver, Detection and Switch Application, which are publicly available[1]. To achieve the required performance, the system offloads the main functionality to the P4-enabled Switch Application, which is the central building block of the data plane. In contrast to other southbound interfaces such as OpenFlow, P4 supports the definition of custom and state-dependent counters, which enable the collection and aggregation of the required metadata. In addition, the Switch Application is configured to emit this data on triggered events via a dedicated port to the Receiver Application, which is located on the control plane within a separated management network for improved security. For every packet sent to the switch by a host, the Switch Application performs the actions as described in Figure 3.

Atfirst the packet is parsed and forwarded to the specific egress port according to a dynamic forwarding table, populated by a simple learning switch. Currently, the application supports headers adhering to the Ethernet protocol, Internet Protocol (IP), Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). In the next step a copy of the packet is created and re-inserted to the processing pipeline. Whereas the original packet is emitted to the specified egress port, the copy is further processed. After creating a hash using a Cyclic Redundancy Check (CRC) algorithm based on the packet's 5-tuple to determine the respective storage registers, the Switch Application checks if the packet is still part of the previousflow. In this case, the registers are updated and the copy of the packet is discarded. Whenever a hash collision occurs, the storedflow is exported, which allows for a trade-off regarding the required memory and the permitted aging offlows. Hence, the recorded metadata of the previousflow is copied to temporary registers and the original registers are re-initialized. At last, a metadata packet is created by overwriting the payload of the copy with the data of the temporary registers and by modifying its destination to sent it to the Receiver Application. In addition, its protocol number is set to 253, which is reserved for experimental purposes by RFC 3692 [28], thus enabling easier identification of metadata packets within the network. As the metadata packet is a valid IP packet it may be forwarded by common switches until it reaches the Receiver Application. In its current state the Switch Application is capable of extracting

[1]https://github.com/lsinfo3/P4FeatureExtraction_HPSR2021

Category							
Counters	**IP**	**TCP/UDP**	**Packet Size**	**TCP RX Wndow Size**	**TCP Control Flags**	**Timestamps**	**Interarrival Time**
NrOfPackets*	SrcAddr*	SrcPort*	Cumulative*	Minimum	NrOfCWR	FirstPacket	Cumulative
FlowDuration*	DstAddr*	DstPort*	Minimum	Maximum	NrOfECE	LastPacket	Minimum
BytesPerSecond*			Maximum		NrOfURG		Maximum
PacketsPerSecond*			Average*		NrOfACK*		Average*
			Standard Deviation*		NrOfPSH		Standard Deviation
					NrOfRST*		LastValue
					NrOfSYN*		
					NrOfFIN*		

TABLE I: Extract-able metadata features and applied subset marked by ”*”.

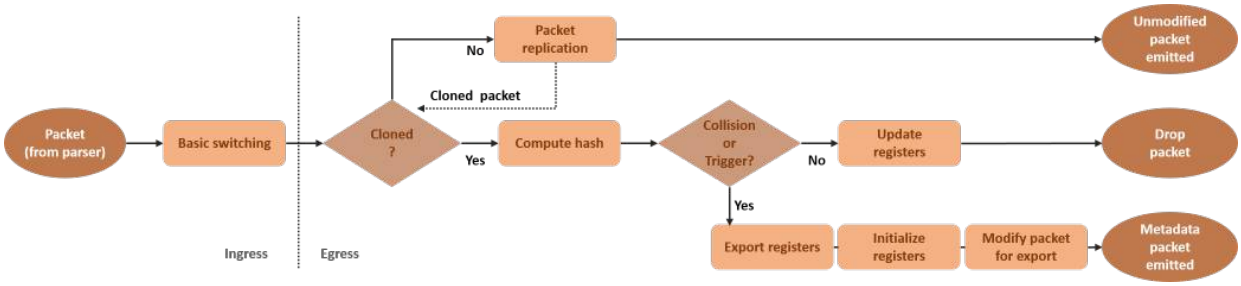

Fig. 3: Programflow of the Switch Application.

a total of 31 uniqueflow-based features, which are summarized in Table I.

As the export of the metadata can only be triggered directly by the Switch Application upon the reception of an incoming packet and is bound to the matchingflow, further mechanisms to initiate an export are needed to guarantee a complete and constant stream of data to the Receiver Application. This task is accomplished by the Controller Application, which utilizes the Apache Thrift API of the Tofino™ platform to read the contents of all registers in regular intervals. It is installed on the x86-based host controller of the switch and polls the registers every 5 seconds. It then derives a list of expired flows, which are defined by having their TCP FIN/RSTflag set or being inactive for at least 45 seconds. The Controller Application then continues to create and send a metadata packet adhering to the same format as described above, before clearing the respective registers. In combination with the trigger-based export done by the Switch Application, the system is able to aggregate the metadata on a perflow basis across multiple packets, hence compressing the amount of metadata which needs to be analyzed.

To enable a complete view of the network and to support multiple Switch Applications, the Receiver Application functions as a central data storage and processing entity. Deployed as a network function within the management network, it runs a DPDK-based application to collect the sent metadata packets. Once a packet is received, the payload containing the metadata is extracted and stored in pre-allocated static data structures to prevent performance impairments due to unnecessary memory operations. This is followed by the computation of additional metadata values for each entry, which are not possible to derive directly on the switch. This includes all calculations containing multiplications and divisions with two dynamic operands, e.g., for computing the averages and standard deviations. Finally, the Receiver Application provides an interrupt routine, which outputs the augmented metadata as CSVfile for further processing.

At last the ML-based Detection Application takes the CSV file of the previous stage as input, hence implementing a producer-consumer-pattern to determine if a given set offlows is benign or malicious. To train the model the Tuesday capture of the CICIDS2017 evaluation dataset [29] was used, as it features realistic and complete traces, which are labeled on a perflow basis and a wide variety of attacks. The model is based on the Random Forest Classifier from scikit-learn, as decision-tree based algorithms have proven to be successful, and performs a pre-processing step on the input. This is done to strip the input from certain features such as the IP address to enable the classifier to determine an attack based on its intrinsic features instead of its origin. To evaluate the model we split the data set into a training and validation subset with a ratio of 75% and 25%, respectively. In addition, we applied a random search algorithm to optimize the hyperparameters of the model by performing a k-fold cross validation using 5 folds. The hyperparameters include the number and depth of the random forests, the amount of features regarded in each split and the quantity of samples in each leaf.

Limitations. Yet, during the implementation certain limitations of the Tofino™ platform had to be overcome. As mentioned, the lack of time-based triggers to export arbitrary registers made the Controller Application mandatory. Although this approach solved the issue of exporting the complete record offlow statistics, it imposes a drawback regarding the timeliness of the data. This is due to the rather poor performance of the only included client, written in Python combined with the Apache Thrift API which averages at 3 seconds for a complete read out of all registers. Hence, this results in a limitation of the polling period of the Controller Application.

In addition, the Tofino™ platform lacks the implementation of certain features, which are defined by the P4 specification. One of these missing features is the ability to truncate a cloned packet. As the metadata is exported via a cloned packet

by overwriting its payload, larger packets than necessary are created whenever the original payload exceeds the space needed for the metadata, hence wasting bandwidth and computational resources of the Receiver Application.

Finally, the limited hardware resources compose the main inhibiting factor. The Tofino™ platform consists of 4 independent pipelines operating in parallel and each having 12 subsequent Match Action Units (MAUs). Each MAU has access to a 10Mbit SRAM block, which is used to store stateful objects. Packet Header Vectors (PHVs) are provided as a data structure to store and process the header and metadatafields. Here, a total of $4096bits$ are available, which are allocated by a block size of 8, 16 and $32bits$. In order to perform arithmetic operations, each MAU may utilize up to 4 Stateful Arithmetic Logic Units (SALUs), which are bound to the respective SRAM block. A SALU may read a stateful object, compare it against a header field or a constant, perform an addition or subtraction as well as store the result. This restricted set of operands is slightly enhanced by the possibility to approximate multiplications with a constant, squares and square roots by using predefined lookup tables. Due to memory constraints, as well as to the limited amount of arithmetic operations to derive theflow based features, the extraction of all features within a single switch is prevented. Therefore, it is required to limit the metadata extraction to a subset of the implemented features, which was selected based on a feature importance analysis.

Furthermore, the computation of more complex features is outsourced to the Receiver Application. As a result the applicability of the system is impaired, as the feature space is limited, which may decrease the accuracy of the Detection Application, while simultaneously increasing the computational requirements of the Receiver Application.

Another side effect of the limited memory is that it increases the load on the Receiver Application indirectly, as it favors hash collisions for rising amounts of concurrentflows, hence resulting in more metadata packets being sent.

At last the Tofino™ platform features $64bit$ timestamps with a nanosecond precision, whereas the SALUs are only able to operate on $32bit$ registers. Hence, to compute features like the Inter Arrival Times (IATs), a system time is emulated by right-shifting the lower $32bit$ by $10bit$ to create a $32bit$ time stamp with microsecond precision, which overflows every 71 minutes. This results on the one hand in a further reduction of the available SALUs and on the other hand in erroneous calculations for long lastingflows.

V. Testbed

To carry out the evaluation, a dedicated testbed was created, which is described in the following. The overall design is in accordance to Figure 2 and relies on a P4-enabled Tofino™ switch. The total switching capacity is specified at $6.5Tbps$, featuring 64 ports each having a maximum capacity of $100Gbps$. In addition, the chassis includes a host controller, which is composed of a 8-core Intel Xenon CPU with $32GB$ of memory running Ubuntu 18.04. Whereas the Switch Application is directly deployed to the switch, the Controller Application is installed to the host controller of the switch. The switch is connected to two servers, each equipped with a 10-core Intel Xenon CPU, a Mellanox ConnectX-5 series NIC with two $100Gbps$ ports and $200GB$ of memory running Ubuntu 18.04. The servers are used to generate/receive traffic as well as to run the Receiver and Detection Application, respectively. For the evaluation of Suricata, the Receiver and Detection Application are replaced by Suricata and the switch is configured to mirror the traffic. Suricata itself is run in the default configuration and the attack specific sections of the Emerging Threats Open [30] are applied as rule set. The traffic is generated by replaying the Tuesday capture of the CICIDS2017 data set, which features port scan, brute force as well as DoS and DDos attacks. The capture consists of $11GB$ spread among 529918 flows and 11609136 packets. As TCPreplay [31] is unable to saturate the $100Gbps$ link, the traces are replayed using the DPDK Burst Replay [32] tool. The entire trace is replayed 10 times, hence resulting in an approximate measurement duration of 90 seconds. To evaluate the impact on the computing resources the servers are monitored by sysstat [33] during the experiments. At last, each experiment is repeated 5 times to account for statistical variances.

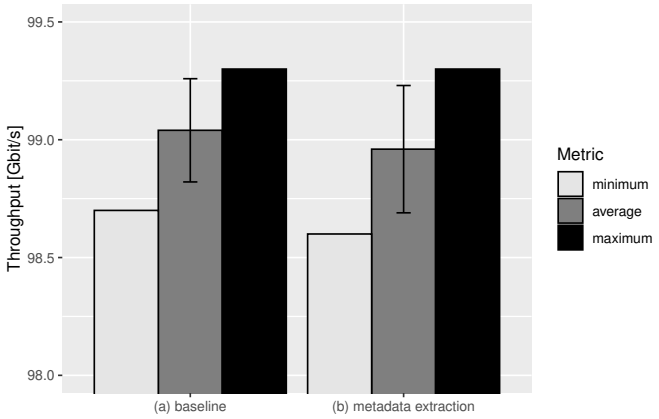

Fig. 4: Throughput of the switch for the baseline (a) and the metadata extraction (b).

VI. Evaluation

In this section we present the results regarding the achievable throughput of the involved system components, the level of obtained aggregation of the traffic as well as a comparison of accuracy with respect to the Detection Application and Suricata. Atfirst the maximum throughput of the Switch Application is examined and we evaluate if the metadata extraction negatively affects the switching capabilities in comparison to a baseline switching application. The results are summarized in Figure 4.

The y-axis displays the achieved throughput in $Gbps$ and the x-axis groups the minimum, average and maximum values of the experiments by the baseline and metadata extraction, respectively. The error bars for the average values denote the 95% confidence intervals. As depicted, both groups are able to operate at $99Gbps$ with slight variations as indicated by the narrow scale of the y-axis, hence fulfilling their tasks

IDS	Suricata				Detection Application			
Label	**TPR**	**TNR**	**FPR**	**FNR**	**TPR**	**TNR**	**FPR**	**FNR**
Benign	98.60%	91.79%	8.21%	1.40%	99.72%	99.85%	0.15%	0.28%
DDoS	2.25%	94.68%	5.32%	97.75%	99.81%	99.98%	0.02%	0.19%
Dos	93.14%	98.03%	1.97%	6.86%	99.62%	99.83%	0.17%	0.38%
Brute Force FTP	99.95%	97.94%	2.06%	0.05%	99.90%	99.99%	0.01%	0.10%
BruteForce SSH	-	-	-	-	99.59%	99.99%	0.01%	0.41%
Port Scan	0.14%	98.75%	1.25%	99.86	99.96%	99.94%	0.06%	0.04%
Weighted Average	84.13%	92.78%	7.22%	15.87%	99.74%	99.86%	0.14%	0.26%

TABLE II: Classification results based on the alerts emitted by Suricata and the Detection Application.

Register Length	Number of Metadata Packets	Traffic Volume
2^16	591329(5.05%)	146.23MB (1.40%)
2^14	693970(5.93%)	174.03MB (1.67%)
2^12	843427(7.20%)	214.15MB (2.05%)

TABLE III: Obtained traffic aggregation levels by the Switching Application. Relative number relate to original data plane traffic.

at nearly line rate. Furthermore, due to the narrow range between the minimum and maximum values both systems provide stable results and a performed t-test resulted in no significant difference between the baseline and metadata extraction measurements.

As the level of traffic aggregation achieved by the Switching Application is correlated to the amount of hash collisions, a parameter study using varying register lengths is performed to emulate the trade off of a higher feature granularity and the supported amount of concurrentflows. The results are presented in Table III. Whereas thefirst column specifies the available register space, the second and third column illustrate the absolute and relative values of the number of metadata packets and their size in respect to the original data plane traffic. As expected the performance of the obtained traffic aggregation decreases if less registers are used. However, in every case a remarkable compression of the original traffic is achieved, hence effectively reducing the load of the downstream system components.

To determine the maximum throughput processable by the Receiver Application, a trace of the generated metadata packets is recorded and TCPreplay with increasing bandwidth limitations is used to determine the point at which packet loss occurs. Unfortunately, the performance of TCPreplay is limited to 7Gbps for the given trace, which induced no packet loss at the Receiver Application. Hence, a maximum bound could not be determined, but taking the achieved aggregation levels into account, the Receiver Application is well able to operate without packet loss for the investigated scenario and does not impose a bottleneck on to the system.

At last the accuracy of the Detection Application is compared to the state-of-the-art IDS Suricata in Table II with regards to the True Positive Rate (TPR), True Negative Rate (TNR), False Positive Rate (FPR) and False Negative Rate (FNR) for the investigated attack categories. As the amount of occurrences varies between the attack categories, hence resulting in an unbalanced data set, the last row addresses this by depicting the weighted average across all attacks. Furthermore, the results for SSH-based brute force attacks have been omitted for Suricata, since the configured rule set featured no support for this attack category. Whereas both systems show good overall classification results, the Detection Application is able to outperform Suricata in all metrics except for the TPR and FNR regarding the brute force FTP attack category. This stems from the fact, that Suricata classifies theflows on a shorter time scale, ideally while they are still inflight. In contrast, the Detection Application operates on larger time frames, hence having more information available. However, during our investigation of Suricata significant packet loss of 9% occurred starting at data plane rates from $600Mbps$, whereas the Detection Application in combination with the Switch, Controller and Receiver Application is capable to perform at data plane rates up to $100Gbps$ without inducing any packet loss, hence being applicable for bandwidth intense core and backbone networks.

VII. Conclusion

Motivated by the increasing demand for network bandwidth and the rising number of attacks on communication networks, this work designed, implemented and evaluated an Intrusion Detection System (IDS), which can be deployed in bandwidth intense networks up to $100Gbps$. To satisfy the high performance demands imposed by the packet processing, the proposed system utilizes a P4-enabled switch as hardware accelerator to offload the extraction of metadata and to aggregate the results. Thus, the load on the downstream systems is reduced, which effectively prevents bottlenecks. The evaluation shows that the extraction of metadata does not affect the capacity of the switch and the application is able to reduce the amount of generated metadata packets down to approximately 5% in comparison to the original data stream. This enables the other system components to process the high traffic demand and the detection accuracy of the proposed system is able to outperform the open source IDS Suricata with respect to the obtained detection accuracy as well as with respect to the achievable throughput. In future work, the model shall be compared with other ML-based IDSs and the functionality may be distributed throughout multiple switches to overcome the limitations of insufficient memory and arithmetic operations to support an enriched feature set.

Acknowledgment: This work has been performed in the framework of the WINTERMUTE project, which is funded by the BMBF (Project ID 16KIS1129). The authors alone are responsible for the content of the paper. The authors want to thank Fabian Biskup for his programming efforts.

REFERENCES

[1] Cisco. Cisco annual internet report (2018–2023). (accessed 2021-03-01). [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

[2] Checkpoint. Security report 2020 | checkpoint. (accessed 2021-03-01). [Online]. Available: https://www.checkpoint.com/downloads/resources/cyber-security-report-2020.pdf

[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, 2020.

[4] C. Kim, A. Sivaraman, N. Katta, A. Bas, A. Dixit, and L. J. Wobker, "In-band network telemetry via programmable dataplanes," in*ACM SIGCOMM*, 2015.

[5] P4 Language Consortium on GitHub. INT Specification Version 2.0. (accessed 2021-03-01). [Online]. Available: https://github.com/p4lang/p4-applications/blob/master/docs/INT_v2_0.pdf

[6] Suricata project. Suricata IDS. (accessed 2021-03-01). [Online]. Available: https://suricata-ids.org

[7] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges,"*IEEE Communications surveys & tutorials*, vol. 18, no. 1, pp. 236–262, 2015.

[8] D. Cerović, V. Del Piccolo, A. Amamou, K. Haddadou, and G. Pujolle, "Fast packet processing: A survey,"*IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3645–3676, 2018.

[9] The P4 Language Consortium. P4 Language Specifications. (accessed 2021-03-01). [Online]. Available: http://p4.org/specs/

[10] Intel Inc. Tofino Series. (accessed 2021-03-01). [Online]. Available: https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch/tofino-series/tofino.html

[11] N. Sultana, N. Chilamkurti, W. Peng, and R. Alhadad, "Survey on sdn based network intrusion detection system using machine learning approaches,"*Peer-to-Peer Networking and Applications*, vol. 12, no. 2, 2019.

[12] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.

[13] Scikit-learn project. Scikit-learn. (accessed 2021-03-01). [Online]. Available: https://scikit-learn.org/stable/

[14] T. Lukaseder, J. Fiedler, and F. Kargl, "Performance evaluation in high-speed networks by the example of intrusion detection,"*arXiv preprint arXiv:1805.11407*, 2018.

[15] S. Zhao, M. Chandrashekar, Y. Lee, and D. Medhi, "Real-time network anomaly detection system using machine learning," in*2015 11th International Conference on the Design of Reliable Communication Networks (DRCN)*. IEEE, 2015.

[16] P. Sangkatsanee, N. Wattanapongsakorn, and C. Charnsripinyo, "Practical real-time intrusion detection using machine learning approaches," *Computer Communications*, vol. 34, no. 18, 2011.

[17] T. N. Thinh, T. T. Hieu, S. Kittitornkun*et al.*, "A fpga-based deep packet inspection engine for network intrusion detection system," in *2012 9th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology*. IEEE, 2012.

[18] Y. Go, M. A. Jamshed, Y. Moon, C. Hwang, and K. Park, "Apunet: Revitalizing {GPU} as packet processing accelerator," in*14th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 17)*, 2017, pp. 83–96.

[19] J. Li, Z. Sun, J. Yan, X. Yang, Y. Jiang, and W. Quan, "Drawerpipe: A reconfigurable pipeline for network processing on fpga-based smartnic," *Electronics*, vol. 9, no. 1, p. 59, 2020.

[20] M. S. Brunella, G. Belocchi, M. Bonola, S. Pontarelli, G. Siracusano, G. Bianchi, A. Cammarano, A. Palumbo, L. Petrucci, and R. Bifulco, "hxdp: Efficient software packet processing on {FPGA} nics," in*14th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI}20)*, 2020, pp. 973–990.

[21] S. Ibanez, G. Brebner, N. McKeown, and N. Zilberman, "The p4-> netfpga workflow for line-rate packet processing," in*Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2019, pp. 1–9.

[22] B. Turkovic, J. Oostenbrink, F. Kuipers, I. Keslassy, and A. Orda, "Sequential zeroing: Online heavy-hitter detection on programmable hardware," in*2020 IFIP Networking Conference (Networking)*. IEEE, 2020.

[23] V. Sivaraman, S. Narayana, O. Rottenstreich, S. Muthukrishnan, and J. Rexford, "Heavy-hitter detection entirely in the data plane," in *Proceedings of the Symposium on SDN Research*, 2017.

[24] M. Dimolianis, A. Pavlidis, and V. Maglaris, "A multi-feature ddos detection schema on p4 network hardware," in*2020 23rd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*. IEEE, 2020.

[25] F. Musumeci, V. Ionata, F. Paolucci, F. Cugini, and M. Tornatore, "Machine-learning-assisted ddos attack detection with p4 language," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020.

[26] 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)*, 2019.

[27] J. Sonchack, A. J. Aviv, E. Keller, and J. M. Smith, "Turboflow: Information richflow record generation on commodity switches," in *Proceedings of the Thirteenth EuroSys Conference*, 2018.

[28] Internet Engineering Task Force. RFC 3692. (accessed 2021-03-01). [Online]. Available: https://tools.ietf.org/html/rfc3692

[29] Canadian Institute for Cybersecurity. ISCX 2012 dataset. (accessed 2020-09-01). [Online]. Available: https://www.unb.ca/cic/datasets/ids.html

[30] Emerging Threats. Emerging Threats rule documentation. (accessed 2021-03-01). [Online]. Available: https://doc.emergingthreats.net

[31]*appneta*on GitHub. tcpreplay project. (accessed 2021-03-01). [Online]. Available: https://github.com/appneta/tcpreplay

[32] *FraudBuster*on GitHub. dpdk-burst-replay project. (accessed 2021-03-01). [Online]. Available: https://github.com/FraudBuster/dpdk-burst-replay

[33] *sysstat*on GitHub. sysstat project. (accessed 2021-03-01). [Online]. Available: https://github.com/sysstat/sysstat