Advancing Network Monitoring with Packet-Level Records and Selective Flow Aggregation

Ina Berenice Fink*, Ike Kunze*, Pascal Hein*, Jan Pennekamp*, Benjamin Standaert§, Klaus Wehrle*, and Jan Rüth*
**Communication and Distributed Systems*, RWTH Aachen University, Germany
§*Washington University in St. Louis*, Missouri, United States
{fink, kunze, hein, pennekamp, wehrle, rueth}@comsys.rwth-aachen.de · b.g.standaert@wustl.edu

Abstract—**Due to its superior efficiency, network operators frequently prefer flow monitoring over full packet captures. However, packet-level information is crucial for the timely and reliable detection, investigation, and mitigation of security incidents. Currently, no solution effectively balances these two contradicting approaches, forcing network operators to compromise between efficiency and accuracy. In this paper, we thus propose HybridMon, a hybrid solution that combines condensed packet-level monitoring with selective flow-based aggregation to strike a new balance between efficiency and accuracy. Operating on the data plane of P4-programmable switches, HybridMon enables fine-grained, practical, and flexible network monitoring at Tbps speeds. We validate the effectiveness HybridMon through extensive evaluations using Internet backbone and university campus traffic traces, demonstrating its reliability and performance in network forensics and intrusion detection contexts. Our results show that HybridMon reliably monitors all flows while reduing the output bandwidth to 12 % to 20 % compared to packet monitoring when exporting standard features. Complementing traditional flow monitoring, we further demonstrate HybridMon's potential to enhance intrusion detection.**

Index Terms—**Security Services, Control and Data Plane Programmability, Monitoring and Measurements**

I. Introduction

Network monitoring serves as a vital data source for a myriad of applications [1], including *network forensics* [2], [3] and *intrusion detection* [4]. However, their operation is challenged by growing network sizes, increasing the vulnerability surface [5], and performance demands of monitoring solutions [6]. Simultaneously, cyberattacks become more sophisticated with deployments, scaling to state-level threats—visible in the Russian invasion of Ukraine [7]—and tailored to individual targets as became evident during the COVID-19 pandemic with contextual phishing and precise attacks against health agencies and hospitals [8]. Therefore, today's network monitoring needs to fulfill two key requirements for effective network security: (i) the ability to handle high traffic volumes and (ii) the timely provision of comprehensive information for detailed analysis.

Two major monitoring approaches exist: packet and flow monitoring. Packet monitoring captures full packets, ensuring maximum accuracy; however, today's traffic volumes render it largely impractical due to exceptional storage and processing demands [9]. Flow monitoring poses an efficient alternative by storing flow statistics in structured records [10]. While it reduces the load on network and subsequent storage and processing components, it nowadays finds increasing use for intrusion detection [11]–[14]. However, relying on flow monitoring for network forensics and intrusion detection limits capabilities and effectiveness due to its significantly lower accuracy [9], [15]–[18], and delayed information availability [14].

Thus, network operators require new reliable approaches that better balance the trade-off between accuracy and performance. To this end, programmable networking devices provide new opportunities for implementing efficient and flexible monitoring [1]. Nonetheless, a broadly applicable solution which balances accuracy and efficiency is still missing, with existing work primarily focusing on performant network telemetry [19]–[25] and traffic monitoring [6], [26]–[28], enhanced flow export [29]–[32], or specific (integrated) security applications [14], [15], [33]–[36].

To close this gap, we propose **HybridMon**, a hybrid approach between packet and flow monitoring that leverages programmable P4-switches. Instead of full packets or aggregated statistics, HybridMon exports a condensed and structured record with *selected packet-level information* for every packet. This approach minimizes overhead while preserving packet-level accuracy, allowing for timely and detailed analysis. Additionally, HybridMon incorporates multiple mechanisms to reduce output and alleviate the load on subsequent components without broadly decreasing accuracy. To this end, *selective aggregation* enables the less accurate flow monitoring for less-relevant (user-defined) shares of traffic, while *fine-granular filtering* allows irrelevant traffic to be excluded upfront. Intrusion Detection Systems (IDSs) can also automatically feed back temporary filter rules, e.g., to exclude malicious high-volume traffic from monitoring once detected. Lastly, HybridMon offers high *deployability* as it runs on commercial-off-the-shelf hardware and can either replace existing switches in-line or be deployed off-path. It handles up to 3.2 Tbps of input per device and provides standardized output compatible with common analysis tools (e.g., nfdump [37]).

Contributions. Our main contributions in this paper are:

- We identify the limitations of traditional network monitoring for various network applications, such as intrusion detection, and derive concrete requirements to fill the gap.
- Addressing these requirements, our P4-based hybrid ap-

proach **HybridMon** efficiently exports customizable packet and flow information in the common IPFIX format.

- Our extensive evaluation of HybridMon demonstrates that operators can benefit from (i) accurate and custom output for security-related tasks, (ii) reliable monitoring, even with heterogeneous traffic patterns, and (iii) increased attack detection compared to traditional flow monitoring.

Open Science Statement. We open-source our implementation [38] under the GPLv3 license.

II. Traditional Network Monitoring and the Road Ahead

The reliability and success of network security applications directly depend on the quality and quantity of their input [39], provided by monitoring systems. In this context, we first examine the trade-offs between packet- and flow-based monitoring. We then derive requirements for the design of new solutions.

A. Comparison of Packet and Flow Monitoring

We assess the content, use for attack detection, and performance of today's prevalent approaches in the following.

Content. While packet monitoring involves full packet capture, flow monitoring exports aggregated statistics of multiple packets from the same flow in the form of *flow records*, which are gathered by *flow collectors* [10]. As a result, flow records lead to the loss of individual packet characteristics. Furthermore, they typically include only information up to the transport layer. The structure of flow records and their detailed transmission are defined by flow export protocols such as NetFlow [40] or IPFIX [41] which serve as input for a broad range of general-purpose analysis tools [37], [42], [43].

Attack Detection. Packet monitoring offers maximum information gain and allows for arbitrary processing, including deep packet inspection (DPI), which facilitates flexible analysis of all (unencrypted) content up to the application layer [44]. Thus, packet monitoring provides the best start for attack detection. In turn, flow monitoring is well applicable for detecting numerically striking, e.g., volumetric, attacks, but is generally less effective at identifying more subtle threats, such as slow or semantic attacks [9], [15]. These include low-rate DoS attacks characterized by periodic short bursts of traffic but low average volume [45]. Recent research further indicates that the performance of ML-based intrusion detection can suffer from the coarse granularity of flow-based statistics [16], [17] while packet-level information, such as packet sizes or inter-arrival times, can significantly enhance the accuracy of novelty detection [18]. Lastly, aggregating packets into flows delays the forwarding of monitored data, potentially resulting in delays of several minutes in attack detection [14].

Performance. Depending on the network, packet monitoring is not always feasible [9] due to its storage and processing requirements. Additionally, resources may be wasted if packets lack usable payloads due to encryption. Flow monitoring substantially reduces the load on subsequent network, storage, and processing components compared to packet monitoring [10], but it requires additional processing power for flow metering and export. Furthermore, flow-based monitoring solutions struggle with short-lived flows [9] and high volumes of minimum-size packets [46], risking further information loss.

Given the shortcomings of packet and flow monitoring, network operators would significantly benefit from hybrid approaches that address their needs.

B. Toward Hybrid Network Monitoring

Based on the weaknesses of packet and flow monitoring, we derive precise requirements for hybrid solutions:

R1 **Deployment:** Typically, network operators already have elaborate network infrastructures and analysis pipelines in use in which new monitoring systems should easily integrate. Providing output in a standardized and structured format, e.g., via IPFIX flow records, facilitates universal use and seamless processing at subsequent components.

R2 **Output:** Accurate packet-level information is critical for attack detection and information above the transport layer can offer additional value [11], [47]. Therefore, solutions should encompass header fields of individual packets and, if unencrypted, customizable application layer information such as request types or error codes depending on the network and its applications. Furthermore, timely provision of the monitored information to security applications is vital for fast attack detection and reaction [14].

R3 **Flexibility:** Reliable operation of the monitoring system and dependent security applications necessitates support of high traffic volumes and abnormal traffic patterns as well as optimized selection of monitored traffic to decrease load. Adjustment of the monitored traffic at runtime allows to proactively relieve all components from benign [48] and adverse traffic after detection [14].

Next, we discuss related work in light of **R1**-**R3**.

III. Related Work

Most related work targets the field of general-purpose network monitoring and telemetry, focussing on performance. Only a subset specifically addresses network security, e.g., by tightly coupling monitoring to security applications.

General Network Monitoring. Various approaches aim to enhance network telemetry through hardware acceleration [19]–[24], [26], [27], [49]. However, these approaches provide network characteristics such as bandwidth and queue times, not traffic information. Solutions for traffic monitoring either seek to relieve existing flow exporters [29], [30] or offer efficient flow export implementations on programmable switches [31]. Consequently, these approaches either maintain or further decrease flow monitoring accuracy. In contrast, Castanheira et al. [50] utilize programmable switches for custom monitoring with flow and packet metrics but restrict it to heavy hitters to reduce load. Sonchack et al. [6] present a hybrid format that combines flow information and packet-level metrics, implementing stateful monitoring on programmable switches. However, they still collect information from multiple packets, delaying its availability and significantly limiting the monitored features due to constrained data plane storage.

Network Security Monitoring. In this emerging research direction, several approaches employ hardware-accelerated monitoring to relieve or enhance security applications, e.g., [15], [33], [34], [51], but either target specific (integrated) applications or only export aggregated information. In contrast, Doriguzzi-Corin et al. [28] extract and store packet-level features for intrusion detection on the data plane of programmable switches and periodically export them to the control plane, but their approach suffers from the same disadvantages as [6].

Implemented in software, Velan et al. [36] complement flow export with event detection, and Hofstede et al. [14] integrate an IDS into a flow exporter to reduce the delay until the IDS can process the monitored information and exclude malicious traffic from monitoring after its detection. Enhancing universal flow monitoring, Erlacher et al. [32] extend IPFIX-based flow records with HTTP-related information. While these solutions modify flow monitoring toward network security, they do not increase its accuracy and cannot handle high traffic volumes.

Overall, research shows that custom monitoring of Tbps throughputs is possible with programmable network devices and can enhance specific security applications. Still, in light of more sophisticated threats, there is a lack of flexible approaches that provide standardized (**R1**) and fine-grained (**R2**) output, and can handle today's traffic volumes (**R3**). To bridge this gap, we propose HybridMon.

IV. Packet-Level Monitoring with Selective Flow-Based Aggregation

This section introduces **HybridMon**, a performant and practical hybrid solution for network monitoring. While hybrid monitoring could be achieved using existing software-based flow exporters like YAF [52], HybridMon leverages P4-programmable switches to achieve flexible monitoring at Tbps speeds, as highlighted by related work. HybridMon specifically targets the Intel Tofino [53] switch to optimize compatibility and functionality with today's state-of-the-art hardware. Still, the design principles of HybridMon are not confined to Intel Tofino and can be adapted to other P4 targets.

We cover the requirements from Sec. II-B as follows:

Deployment. Fulfilling **R1**, our monitoring relies on established protocols for flow export, ensuring compatibility with common analysis tools. Running on a single switch, HybridMon is readily deployed in networks in-line or off-path.

Output. In line with **R2**, HybridMon instantly exports flow records with a flow size of 1, i.e., one record for every packet, to deliver packet-level information while covering a comprehensive and customizable range of features. Thus, HybridMon facilitates rich, time-critical analysis.

Flexibility. Addressing **R3**, HybridMon includes a traffic filter and allows for the aggregation of selected (low-interest) traffic into flow statistics to reduce output. It further operates entirely on the switch's data plane, avoiding CPU-based bottlenecks and enabling reliable handling of diverse traffic patterns.

Design Overview. HybridMon comprises three (data-plane) components illustrated in Fig. 1. First, the ① Packet Filter identifies relevant traffic and copies the respective packets

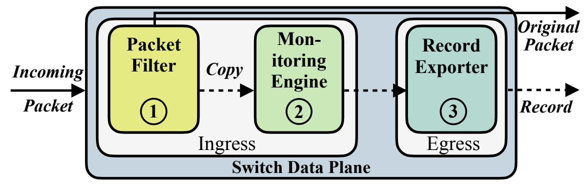


Fig. 1. HybridMon's three components: ① The packet filter extracts features, sorts out irrelevant traffic, and copies relevant packets. These copies are further processed by ② the monitoring engine before ③ the record exporter outputs flow records, either containing packet-level information or flow statistics.

to the ② Monitoring Engine, which determines whether to subsample the corresponding flow. Then, the packet copy is passed to the ③ Record Exporter, which probabilistically generates a flow record for subsampled flows and a packet-level record for each packet in unsubsampled flows. The original packets remain unmodified and, if HybridMon is deployed in-line, are forwarded to their destination.

The detailed design of HybridMon's three components is depicted in Fig. 2 and presented in the next subsections.

A. Packet Filter

The packet filter operates in the ingress of the switch and extracts all relevant packet features from incoming packets. It then uses match-action tables to exclude specific traffic from monitoring. By filtering out irrelevant or volumetric attack traffic (once detected by an IDS), we conserve processing resources on our switch and subsequent network components. While the filter features, e.g., IP addresses or protocol, are statically defined, the respective match values can be dynamically inserted at run-time to enable quick adjustments to the monitored traffic during operation. If no filter applies, a copy of the packet is forwarded to the monitoring engine.

B. Monitoring Engine

By default, HybridMon exports packet-level records, but the monitoring engine also supports selective subsampling, i.e., statistic-based aggregation of multiple packets into a single flow record. Operators can define which flows to subsample at run-time using a match-action table, based on factors like origin, destination, or protocols. If a packet does *not* match the user-defined rules, we let the record exporter ⓐ generate a packet-level record solely based on that individual packet. To achieve subsampling and export of flow statistics for subsampled flows, the monitoring engine relies on two key mechanisms: *flow tracking* and *probabilistic export decision*.

1) Flow Tracking: To track subsampled flows, we use a *flow register* in the ingress, indexed by hashes of flow information tuples. These tuples include protocol, source/destination IP addresses (IPv4 and IPv6), and source/destination ports or ICMP type/code. To detect collisions, we additionally store the tuples of the already monitored subsampled flows at their respective indices and compare them with the tuple of incoming packets using the following logic:

Empty Entry. If the register entry is empty, the packet is ⓑ recirculated to insert the flow tuple.

Collision. If the entry is not empty but the stored tuple does not match the tuple of the incoming packet, a collision occurs.

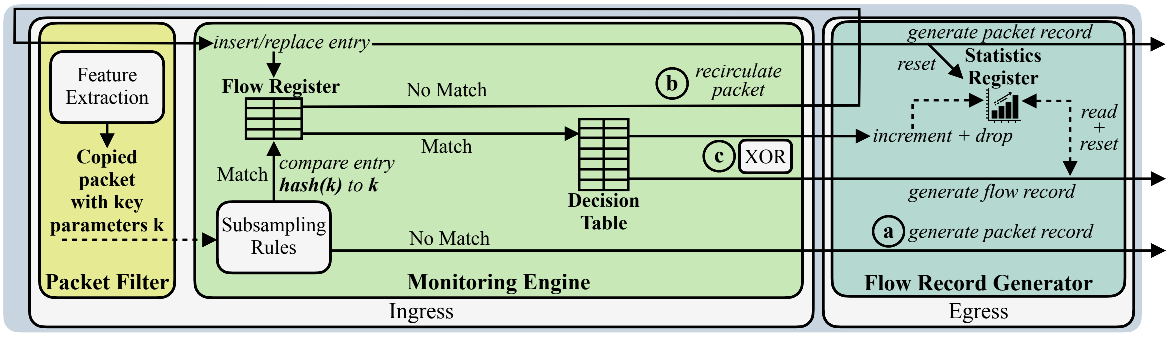


Fig. 2. Our monitoring engine uses copies of the original packet and maintains rules to decide whether to subsample a flow using a flow register that tracks actively monitored flows. Packets *not* belonging to subsampled flows always trigger ⓐ the generation of packet-level records. The first packet of each new subsampled flow is ⓑ recirculated to insert an entry into the tracking table before we create a first packet-level record. Based on a decision table, packet copies of already monitored subsampled flows are ⓒ either dropped in the egress after updating the statistics or trigger statistics-based flow record generation.

This will, at the latest, happen if the number of subsampled flows exceeds the flow register's capacity In this case, we can either monitor the new flow at packet-level granularity until the register entry is freed or ⓑ recirculate the packet analogously to replace an existing flow. However, due to our design, we can only create flow records for *incoming* packets and not trigger record generation at an arbitrary time. Thus, replacing a flow may result in losing its accumulated statistics since its last export. We present multiple mechanisms to mitigate the effect of this limitation in Sec. IV-B2.

Match. If the entry is not empty and matches the tuple of the incoming packet, the flow is already monitored. In this case, we probabilistically decide whether to export a *flow record*.

2) Probabilistic Export Decision: For subsampled flows, we probabilistically generate flow records based on accumulated statistics. These flow statistics are stored in a dedicated *statistics register* in the egress and are reset after each export. The export decision on whether to generate a record for a flow is based on the probability $p_{record}(s) \approx \min(1, \frac{k}{s})$, where s is the flow size k (i.e., the flow encompasses s packets) and k is a tunable parameter. We model p_{record} using a match-action table as described in [54]. Since we can only approximate p_{record} as Tofino, like most ASIC switches, provides only integer arithmetic logic units, setting k to a power of two simplifies these approximations.

Loss Mitigation. A larger k increases the rate of flow records, reducing the time interval between two records and the information loss when flows are replaced. Furthermore, due to its logarithmic behavior, p_{record} is higher for small flows, limiting information loss, while large flows generate fewer records, keeping the total record count at a moderate level.

To prevent loss and cover every subsampled flow in our output in case its flow register entry is replaced early, we always generate a flow record for a flow's first packet. Additionally, we generate a record if a final packet of a TCP flow comes in, which we identify via `TCP RST` and `TCP FIN` flags.

Enforcement. Having decided whether to generate a record nor not, we use ingress-to-egress mirroring to forward a copy of the packet to HybridMon's third component, the record exporter. We signal all necessary actions using the target port of the mirrored packet and a dedicated mirror header field.

C. Record Exporter

Once the decision on whether to generate a record is made, the *record exporter* implements the corresponding action in the egress pipeline. In addition, flow collectors also expect periodic *template records* [40], [41], which specify the record structure and interpretation.

1) Template Records: We leverage Tofino's packet generator on the control plane to create static template packets at the desired periodicity, which are forwarded to the collector by the data plane. The record exporter stores the ID of the last-received template in a register, enabling the association of subsequent records with that template. We further employ templates with a custom control header to periodically piggyback control information from the control plane, e.g., the collector's address or the Unix time to realize accurate timestamps.

2) Packet-Level and Flow Records: If no record is needed, the flow exporter ⓒ updates the statistics of the respective flow and drops the packet. Otherwise, it ⓐ crafts packet-level records using features extracted in the ingress or ⓒ flow records based on the subsampled data from the egress' *statistics register*. The generation of all packet and flow records is fully bound to the data plane due to our performance demands. However, the core P4 language does not provide mechanisms to create *new* arbitrary packets on the data plane. Therefore, we exploit the traffic packets mirrored from the ingress:

First, we extract all relevant information, then skip the irrelevant remainder of the packet. To create records, we add predefined headers representing IP, UDP, and the flow export protocol, i.e., IPFIX or NetFlow, to the packet. These headers contain information such as the collector's address, the latest template ID, timestamp, and sequence number. An additional header structure represents the record payload, which is filled with the extracted packet-level information and/or monitored statistics. Lastly, the newly crafted record packet is exported to the predefined collector for further processing.

Overall, HybridMon enables detailed and customizable inline and off-path network monitoring in high-speed networks, addressing the needs identified in Sec. II-B. In the remainder of this paper, we demonstrate its feasibility by presenting and evaluating an implementation for Tofino1.

V. Open-Source Implementation of HybridMon

We implemented HybridMon in P4 for the Intel Tofino1 on top of basic Longest Prefix Match (LPM) routing functionality. We chose the IPFIX protocol [41] as it is a widely supported open standard, increasing deployability. In the following, we discuss our implemented subsampling and monitored features.

A. Subsampling Strategy

Heavy hitters are prime candidates for subsampling as packets of these flows typically exhibit similar characteristics, making their aggregation less critical regarding information loss while providing significant reduction potential. Thus, our monitoring engine implementation targets heavy hitters by leveraging PRECISION [54] with a 2-way associative flow table, split over 2 register arrays, and simultaneously serving as flow register as described in Sec. IV-B1 and as heavy hitter list. We configured a flow table size of 65 536 entries, which is the maximum share of subsampled flows supported by our implementation on the used hardware. We further lowered the recirculation probabilities in our implementation by a factor of 6 compared to PRECISION, increasing the monitoring capacity. Last, we set $k = 32$ as default parameter for p_{record} (cf. Sec. IV-B2), enabling a reasonable trade-off between export frequency of subsampled flows and accuracy.

B. Monitored Features

We currently support the extraction of 27 standard IPFIX Information Elements (IEs) defined by IANA [55], including source and destination MAC and IP addresses, protocol ID, source and destination ports, number of octets, flow start time, IP time-to-live, fragmentation offset, ID, and flags, ICMP type and code, TCP flags, sequence number, and window size, and HTTP status code. Employing enterprise-specific IEs [55], we further introduced port-based detection of startTLS, and DNS over UDP, including DNS request and response code. These examples demonstrate that HybridMon can easily export custom flow and packet-level features (cf. **R2**) above the transport layer while fully complying with the IPFIX protocol. In total, our implementation currently supports 31 features, and our tests showed that we can easily employ 10 counters of 4 B each for applicable features to keep the state of subsampled flows without memory issues.

Our prototype demonstrates that HybridMon can be implemented on off-the-shelf switch hardware and supports standard flow export protocols, thereby satisfying **R1**. Subsequently, we evaluate our implementation with respect to **R2** and **R3**.

VI. Evaluation of HybridMon

We provide an extensive evaluation to prove the feasibility of our design and ensure that the requirements established in Sec. II-A are met. We first investigate the data quality of HybridMon's output in Sec. VI-A and how it copes with traffic from Internet backbone links in Sec. VI-B, comparing it to the open-source flow exporter YAF [52]. Afterward, we assess HybridMon's effect on attack detection and back-feeding of filter rules in Sec. VI-C. Last, we examine its performance and benefits in a university-specific use case in Sec. VI-D.

A. Data Quality Assessment

We first examine the data quality of HybridMon to ensure that it is sound and can be used as input for reliable analysis (**R2**). For this purpose, we generated IPFIX records with and without HybridMon's heavy hitter-based subsampling, i.e., a mix of packet-level and flow records vs. packet-level records only. We compared the output to (i) the input and (ii) the traditional IPFIX-based flow records exported by YAF [52]. To highlight the applicability to production traffic, we used four real-world traffic traces as input, each randomly chosen from the publicly available datasets provided by CAIDA [56]–[59], containing extensive anonymized packet captures with real traffic recorded at high-speed backbone links.

To evaluate HybridMon on the datasets, we connected two workstations *W1* and *W2*, using 10 Gbps links to one 32-port Tofino-based switch running HybridMon. Then, we replayed each trace from W1 to the switch and collected its output, i.e., the generated IPFIX records, at W2. We conducted 10 runs for each of the four CAIDA traces to obtain significant results. Due to processing limitations of our workstations, we replayed the CAIDA traces at 100 Mbps and fed the same slowed-down traces into YAF to obtain comparable output. This adaption does not influence our evaluation results as it only affects the packets' timestamps but not the metadata, such as IP addresses or protocols (i.e., the targets of our evaluation).

1) Packet Coverage: We first evaluated the share of packets reported for every {protocol, direction, port}-tuple. To this end, we counted the packet numbers for each tuple in the original traces and compared them against the records generated by HybridMon and YAF. YAF covered 100 % of the packets of each tuple in its records. The same was the case for HybridMon *without* subsampling. With subsampling, HybridMon only reported 92 % of a tuple's packets in the worst case due to replacement-based statistic losses (cf. Sec. IV-B1). However, for 99.9 % of the tuples, the package coverage was still above 95 %, and 83 % of the tuples had 100 % of their packets reported.

2) Flow Coverage: Flow-based network monitoring typically captures connections defined by a five-tuple of source and destination IP address, source and destination port, and protocol. Thus, every flow that occurs in the traffic must also be contained in the monitored data. We listed for each input trace which flows it contains and analyzed which flows were covered by the output records of HybridMon. The results showed that depending on the CAIDA trace, only 1 to 3 flows were not captured while each CAIDA trace contained between 1M and 3M flows in total. However, as no packet loss was observed before, we conducted a closer investigation and found that none of the flows was actually lost but wrongly labeled for one of two reasons: (a) IP-in-IP encapsulation is currently not supported by our implementation, and (b) ICMPv6 fragments are cut off in the CAIDA traces due to anonymization. In both cases, our parser does not detect transport protocol information, setting the ports of the captured flows to 0. Our implementation can be easily extended to cover the first case,

TABLE I
SHOWN ARE THE OUTPUT QUANTITIES AS A SHARE (IN %) OF THE INPUT QUANTITIES OF THE DIFFERENT APPROACHES AND HYBRIDMON'S SUBSAMPLING RATE, I.E., HOW MANY FLOWS WERE SUBSAMPLED. IPFIX RECORDS ARE RELATIVE TO THE INPUT PACKET COUNT.

Trace	Method	Packets	Bytes	IPFIX Records
CAIDA2011A	HybridMon	100.00	20.22	100.00
	sub. (7.21%)	51.42	10.40	51.42
	YAF	4.89	1.16	12.09
CAIDA2011B	HybridMon	100.00	17.61	100.00
	sub. (11.99%)	41.23	7.26	41.23
	YAF	2.78	0.70	8.52
CAIDA2015A	HybridMon	100.00	16.91	100.00
	sub. (8.15%)	46.25	7.80	46.24
	YAF	1.28	0.42	4.99
CAIDA2018A	HybridMon	100.00	12.07	100.00
	sub. (8.34%)	37.20	4.50	37.19
	YAF	3.57	0.43	7.12

and the second case would not occur with live traffic. Thus, the results indicate full coverage and reliable monitoring of all flows by HybridMon with and without subsampling.

B. Monitoring Efficiency and Throughput

Next, we evaluated the resource efficiency and monitoring capacity of HybridMon (**R3**) using real hardware.

1) Resource Efficiency: We first evaluated HybridMon's output size in terms of packets, bytes, and records, again comparing it against the open-source tool YAF. To this end, we generated records including standard flow features (i.e., TCP flags and packet and byte counters), resulting in records of 50 B for HybridMon and 49 B for YAF. We then conducted 10 runs for each of the four CAIDA traces and averaged the numbers of exported packets, bytes, and records (cf. Tab. I).

YAF applies advanced software-based aggregation to *all* flows, creating low record numbers of up to 12 % of the input packets. YAF can also aggregate multiple records into one record packet, leading to even lower numbers of record packets than records and only 0.4 % to 1 % of the original bandwidth. In contrast, HybridMon's implementation on switch hardware does not enable such record aggregation. Thus, each record requires a packet and the number of output packets always equals the number of output records. Logically, no reduction of the output records and packets occurs with deactivated subsampling, where one packet-level record is generated for every monitored packet. Still, even this 1-to-1 mapping significantly reduces the bandwidth, as the condensed IPFIX records only need 12 % to 20 % of the original bandwidth. In turn, heavy hitter-based subsampling, which affected 7 % to 12 % of the flows, reduced the number of output records and record packets by around 48 % to 63 %, reducing the bandwidth to 4.5 % to 10.5 % of the original trace.

The results of this evaluation show that HybridMon provides lower output efficiency compared to traditional flow monitoring, which is an expected consequence of its higher granularity. However, its bandwidth is significantly reduced compared to packet monitoring, even without aggregation. We thus conclude that HybridMon offers a reasonable trade-off between accuracy and efficiency, which is even tunable by adapting the share of subsampled traffic. Furthermore, we did not deploy any filter rules for our evaluation, which will additionally reduce the output size. Last, software-based solutions, e.g., deployed at additional middleboxes, could complement HybridMon by providing subsequent aggregation of its output and further reducing the load on the network.

2) Monitoring Capacity: Our Tofino1 switch has two independent pipelines, each accommodating 16 ports. The monitoring engine in our implementation leverages packets received on a single pipeline. Separate monitoring with both pipelines is feasible, e.g., to monitor different subnets, and theoretically allows for up to 3.2 Tbps for off-path monitoring and up to 1.6 Tbps for in-line monitoring. However, in practice, there are additional factors to consider, particularly the impact of recirculation when using subsampling, the deployment scenario, and different input patterns such as attack traffic. We evaluate the effects of these factors subsequently.

Impact of Recirculation. Our implementation employs heavy hitter-based subsampling using PRECISION [54], which probabilistically recirculates a portion of the input packets to add them to the flow register/heavy hitter list. This recirculation can reduce the amount of traffic HybridMon can handle to less than the switch's maximum capacity. To measure the impact of recirculation, we replayed each of the four CAIDA traces 10 times as done in Sec. VI-A and counted the recirculated packets. Recirculation rates for these traces were consistent across runs and ranged from 0.6 to 1.1 %. We also measured the recirculation rate for the measurement traffic described in Sec. VI-D, resulting in 2.8 %. These results show that recirculation has only minimal impact on the switch's capacity. With average traffic, Tofino1 with 3.2 Tbps line-rate can support around 3.16 Tbps. Even subject to unusual traffic, where the heavy hitter table is ineffective, recirculation only reduces the switch's maximum capacity to around 3.1 Tbps.

Impact of Deployment. The actual monitoring capacity further depends on the deployment scenario. To forward records directly to connected collectors, i.e., without loading the rest of the network, we need to occupy dedicated collector ports. Furthermore, in the case of short flows with small packets, e.g., caused by DNS traffic or SYN flooding, record packets might be more than twice the size of the original packet since subsampling cannot be applied and current data plane capabilities do not allow exporting multiple records in one packet. To prevent record loss in this worst case, we need to account for twice as much output as input, dedicating at least twice as many ports to the collector ports as to the incoming monitored traffic. Then, HybridMon provides a monitoring capacity of up to 1.03 Tbps for in-line and off-path monitoring when using both pipelines and already considering recirculation. Thus, even in the worst case, HybridMon offers a significantly higher monitoring capacity than commercial hardware appliances for flow monitoring, which typically support up to 100 Gbps per device, e.g., the Flowmon Probe [60].

Impact of Volumetric Attacks. Just as other monitoring approaches, HybridMon is not immune to volumetric attacks. In particular and as described above, short flows with small packet sizes can amplify the monitoring output. However, HybridMon is more robust than common approaches that aggregate records on the control plane as their performance depends heavily on the successful aggregation of packets, which fails in face of many simultaneous small flows. In contrast, such approaches quit operation when the control plane is overloaded, hindering attack detection. HybridMon keeps the monitoring and subsequenty security applications running as long as the data plane capacity is not exceeded (cf. Sec. VI-B2). Additionally, as soon as an attack is detected, it can be blocked from monitoring by uninvasively installing respective filter rules, as demonstrated subsequently.

C. Detection and Mitigation of Attack Traffic

As HybridMon is designed to provide enhanced input for various applications compared to traditional flow monitoring, we also demonstrate its effectiveness for detecting attacks with IDSs (**R2**) and support for reactive installation of filter rules (**R3**), e.g., for security purposes.

1) Effectiveness for Intrusion Detection: We evaluated the effectiveness of HybridMon and its output for intrusion detection by measuring the detection performance of the Kitsune IDS [61] on raw packets as well as flow and hybrid records. Kitsune originally extracts features from full packet captures and calculates statistics for anomaly detection. To provide significant and comparable results, we repeated Kitsune's original evaluation on Mirai attack traffic, which is available with their demo code [62], but extended Kitsune to also extract and process IPFIX records, complementing its features.

We fed the Mirai traffic trace, which contains ARP flooding, into YAF to export traditional flow records and into HybridMon to export unsubsampled hybrid records, providing the same features but different granularities. In detail, YAF always aggregated around 1500 ARP requests into one flow record while HybridMon generated one individual record for each ARP request. Since we expected a granularity similar to that provided by YAF for subsampled records generated with HybridMon, we did not investigate HybridMon detection performance with subsampling separately at this point.

After record generation, we trained and tested Kitsune on the raw Mirai packets and on the flow and hybrid records using Kitsune's original training configuration, setting the threshold to provide an FPR of 0.001. While the balanced accuracy of flow and hybrid records was similar, the higher granularity of the hybrid records lead to a significantly higher precision and F1 score (cf. Tab. II). Thus, our results indicate that hybrid monitoring as provided by HybridMon can effectively improve the detection rates of security applications compared to standard flow monitoring. However, future research on a broader range of security applications and diverse datasets is needed to evaluate the potential of hybrid monitoring approaches fully.

2) Installation of Filter Rules at Run-Time: Fast and flexible configuration of the monitored traffic, e.g., exclusion of attack traffic after its detection, is critical to maintain operation when facing heterogeneous traffic [48]. We demonstrate this HybridMon feature by implementing a proof-of-concept with

TABLE II
DETECTION PERFORMANCE OF KITSUNE ON THE MIRAI ATTACK TRACE FOR FULL PACKETS VS. OUTPUT OF YAF AND HYBRIDMON.

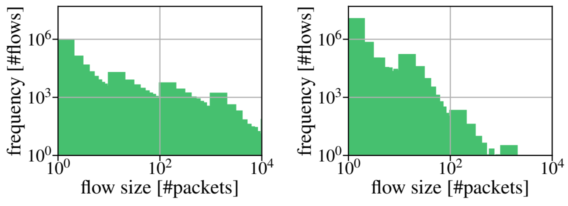
Approach	**Balanced Accuracy**	**Precision**	**F1**
Full Packets	0.939	0.999	0.936
YAF	0.499	0.321	0.486
HybridMon	0.523	0.991	0.923

a simple threshold-based IDS written in Python, deployed at our collector, and the Mirai attack traffic from the Kitsune dataset [61] as input. We configured the IDS to trigger an alert when its threshold is reached and then immediately install a filter rule via HybridMon's control plane using a TCP client/server implementation. We then measured the time between alert and rule deployment over 10 runs and made sure that further traffic from the attacking MAC address was excluded from the hybrid records. Averaged over 10 runs, it took only 1.272 s from triggering the alert at the IDS until the new filter rule was successfully installed. Therefore, our evaluation shows that HybridMon is quickly adapted to sudden events in the network, e.g., triggered by security applications.

D. Use Case Evaluation: University Network Traces

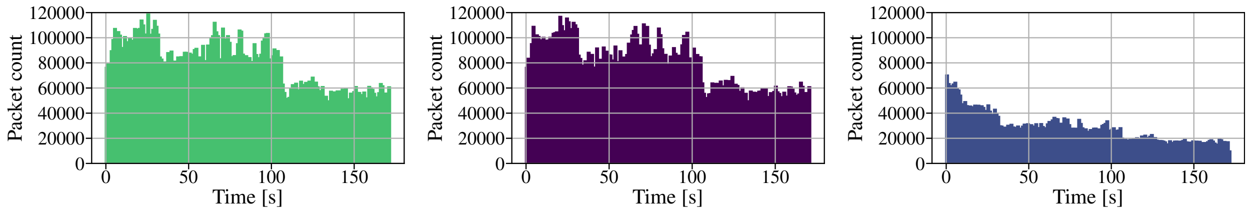
HybridMon is designed to address the needs of network operators who operate a large backbone with heterogeneous network use regarding throughputs, users, and machines. Therefore, we verify the performance of HybridMon in our campus network (**R3**), which we present subsequently. Then, we discuss how HybridMon's functional features benefit this use case (**R2**, **R3**).

1) Performance Evaluation: University networks must both cope with high traffic volumes and unusual traffic patterns due to ongoing research, e.g., caused by Internet measurements at our computer science department [63], resulting in outstanding shares of UDP traffic and small flows. Indeed, our network operators report that commercial flow export solutions, as partly deployed at our university's backbone, at times struggle with handling these loads and fail to generate records. Thus, we evaluated the performance of our HybridMon with real traffic traces covering two scenarios: standard and research traffic. Specifically, we were provided with truncated traffic traces captured at the peering point of RWTH Aachen University with its transit network, i.e., all non-internal traffic traverses this link. The flow size distributions of both traces are provided in Fig. 3a) and Fig. 3b), hightlighting their differences.



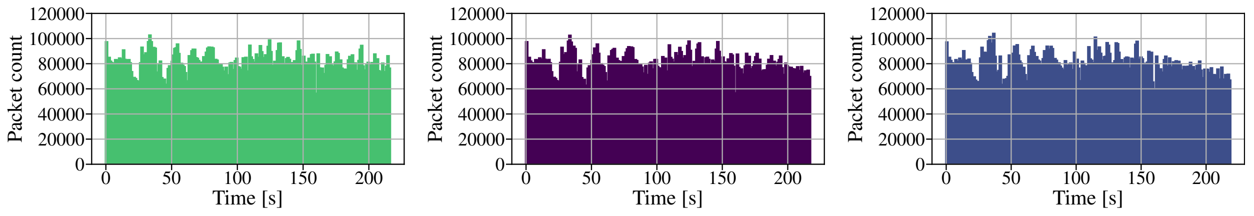
(a) Regular traffic (b) Research traffic

Fig. 3. Flow size distributions of regular and abnormal research traffic captured at our university's backbone.



(a) University traffic (input) (b) HybridMon output (full) (c) HybridMon output (subsampled)

Fig. 4. Packet distributions (1 s bins) of input/output traffic originating from our university network's peering point.



(a) Measurement traffic (input) (b) HybridMon output (full) (c) HybridMon output (subsampled)

Fig. 5. Packet distributions (1 s bins) of input/output traffic originating from our university's data center containing only special measurement traffic.

Regular Operation. The regular traffic capture covers almost 3 min of RWTH Aachen University's full traffic. We visualize its packet count distribution in Fig. 4a). Fig. 4b) details the output of HybridMon without subsampling, showing that the output distribution mirrors the input distribution. In contrast, Fig. 4c) shows the output when applying subsampling, resulting in significantly reduced packet counts.

Research Traffic. We provide the packet count distribution of the examined research traffic in (cf. Fig. 5a)). Using the respective capture as input for HybridMon, we observed that our subsampling strategy had little impact, resulting in similar input and output distributions for full and subsampled flow monitoring (cf. Fig. 5a) and Fig. 5c). However, all flows of the original trace were preserved, again covering over 99.9 % percent of their respective packets on average (cf. Fig. 5b).

Discussion. Similar to the attack traffic discussed in Sec. VI-B2, the research traffic makes it hard to define permanen heavy hitters for subsampling due to its many small flows. Still, this challenging traffic is handled without effort or accuracy loss. Furthermore, our evaluation demonstrates that HybridMon effectively reduces the monitoring output compared to packet monitoring for regular traffic, i.e., the large majority of traffic in our university's network.

2) Functional Benefits: As discussed in Sec. VI-B2, HybridMon can handle challenging traffic as occurring in our university's network with up to 1.03 Tbps. According to our network operators, this throughput and the port density of Tofino1 are more than sufficient to cover the whole traffic of RWTH Aachen University (around 40 Gbps) with a single switch for the foreseeable future. Research traffic is also one example of traffic that regularly occupies high shares of monitoring and processing resources, although it is known and not of interest from a security perspective, and can be reasonably excluded through filtering (cf. Sec. IV). Last, we added a port-based startTLS detection to HybridMon's feature list upon request of our university's network operators, who are particularly interested in observing such connections due to the numerous vulnerabilities of startTLS [64]. This highlights the capability and benefits of HybridMon for monitoring custom features to increase the visibility of critical characteristics.

Overall, our results emphasize that HybridMon can handle real-world traffic without any particular restrictions and provide use case specific features. We plan to thoroughly study HybridMon in a production environment at the university's uplink and replace commercial solutions if applicable.

VII. Conclusion

Network monitoring is key for detecting and investigating attacks, but we identify a lack of balanced hybrid solutions between packet monitoring and flow monitoring. To fill this gap, we propose HybridMon, which combines IPFIX-based packet-level records with custom features and selective flow aggregation. It further allows adapting the monitored traffic at run-time, e.g., as an automated reaction by an IDS. Consequently, HybridMon offers sensitive and resource-efficient monitoring with a tunable trade-off between efficiency and accuracy. Our open-source implementation is readily deployable in-line or off-path and exports records at Tbps while providing compatibility with existing flow-based analysis tools. Our evaluation confirms the practical feasibility of HybridMon for reliable and reactive network monitoring in the wild. We further demonstrate its efficiency gain compared to packet monitoring and its potential to increase the effectiveness of security applications compared to traditional flow monitoring.

Especially as broadly applied encryption (e.g., through QUIC) might eventually render full packet captures obsolete, we expect network operators to have an increasing demand for efficient monitoring of packet-level metadata, for which HybridMon provides an efficient solution. In the future, we will combine HybridMon with diverse analysis tools, e.g., IDS, to further evaluate the impact of its increased monitoring granularity compared to traditional flow monitoring.

References

[1] D. Ding, M. Savi, F. Pederzolli, and D. Siracusa, "Design and Development of Network Monitoring Strategies in P4-enabled Programmable Switches," in *Proceedings of the 2022 IEEE/IFIP Network Operations and Management Symposium (NOMS '22)*, IEEE, 2022.

[2] K. Kent, S. Chevalier, T. Grance, and H. Dang, "Guide to Integrating Forensic Techniques into Incident Response." NIST SP 800-86, 2006.

[3] R. Hunt and S. Zeadally, "Network Forensics: An Analysis of Techniques, Tools, and Trends," *Computer*, vol. 45, no. 12, pp. 36–43, 2012.

[4] B. Mukherjee, L. T. Heberlein, and K. N. Levitt, "Network Intrusion Detection," *IEEE Network*, vol. 8, no. 3, pp. 26–41, 1994.

[5] S. J. Saidi, A. Maghsoudlou, D. Foucard, G. Smaragdakis, I. Poese, and A. Feldmann, "Exploring Network-Wide Flow Data With Flowyager," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 1988–2006, 2020.

[6] J. Sonchack, O. Michel, A. J. Aviv, E. Keller, and J. M. Smith, "Scaling Hardware Accelerated Network Monitoring to Concurrent and Dynamic Queries With *Flow," in *Proceedings of the 2018 USENIX Annual Technical Conference (ATC '18)*, pp. 823–835, USENIX Association, 2018.

[7] European Parliament, "Russia's war on Ukraine: Timeline of cyber-attacks." https://www.europarl.europa.eu/thinktank/en/document/EPRS_BRI(2022)733549, 2022 (accessed January 5, 2024).

[8] H. S. Lallie, L. A. Shepherd, J. R. C. Nurse, A. Erola, G. Epiphaniou, C. Maple, and X. Bellekens, "Cyber security in the age of COVID-19: A timeline and analysis of cyber-crime and cyber-attacks during the pandemic," *Computers & Security*, vol. 105, 2021.

[9] A. Sperotto, G. Schaffrath, R. Sadre, C. Morariu, A. Pras, and B. Stiller, "An Overview of IP Flow-Based Intrusion Detection," *IEEE Communications Surveys & Tutorials*, vol. 12, no. 3, pp. 343–356, 2010.

[10] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow Monitoring Explained: From Packet Capture to Data Analysis With NetFlow and IPFIX," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2037–2064, 2014.

[11] F. Erlacher and F. Dressler, "FIXIDS: A High-Speed Signature-based Flow Intrusion Detection System," in *Proceedings of the 2018 IEEE/IFIP Network Operations and Management Symposium (NOMS '18)*, IEEE, 2018.

[12] L. Hellemons, L. Hendriks, R. Hofstede, A. Sperotto, R. Sadre, and A. Pras, "SSHCure: A Flow-Based SSH Intrusion Detection System," in *Proceedings of the 6th IFIP International Conference on Autonomous Infrastructure, Management, and Security (AIMS '12)*, vol. 7279, pp. 86–97, Springer, 2012.

[13] L. Dias, S. Valente, and M. Correia, "Go With the Flow: Clustering Dynamically-Defined NetFlow Features for Network Intrusion Detection with DynIDS," in *Proceedings of the 2020 IEEE 19th International Symposium on Network Computing and Applications (NCA '20)*, pp. 1–10, IEEE, 2020.

[14] R. Hofstede, V. Bartoš, A. Sperotto, and A. Pras, "Towards Real-Time Intrusion Detection for NetFlow and IPFIX," in *Proceedings of the 9th International Conference on Network and Service Management (CNSM '13)*, pp. 227–234, IEEE, 2013.

[15] S. Panda, Y. Feng, S. G. Kulkarni, K. K. Ramakrishnan, N. Duffield, and L. N. Bhuyan, "SmartWatch: Accurate Traffic Analysis and Flow-State Tracking for Intrusion Prevention Using SmartNICs," in *Proceedings of the 17th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT '21)*, pp. 60–75, ACM, 2021.

[16] H. Clausen, R. Flood, and D. Aspinall, "Controlling Network Traffic Microstructures for Machine-Learning Model Probing," in *Proceedings of the 2021 EAI International Conference on Security and Privacy in Communication Networks (SecureComm '21)*, vol. 398, pp. 456–475, Springer, 2021.

[17] M. A. Salahuddin, M. F. Bari, H. A. Alameddine, V. Pourahmadi, and R. Boutaba, "Time-based Anomaly Detection using Autoencoder," in *Proceedings of the 2020 16th International Conference on Network and Service Management (CNSM '20)*, IEEE, 2020.

[18] K. Yang, S. Kpotufe, and N. Feamster, "Feature Extraction for Novelty Detection in Network Traffic." arXiv:2006.16993, 2020.

[19] A. Gupta, R. Harrison, M. Canini, N. Feamster, J. Rexford, and W. Willinger, "Sonata: Query-Driven Streaming Network Telemetry," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '18)*, pp. 357–371, ACM, 2018.

[20] Y. Zhou, C. Sun, H. H. Liu, R. Miao, S. Bai, B. Li, Z. Zheng, L. Zhu, Z. Shen, Y. Xi, P. Zhang, D. Cai, M. Zhang, and M. Xu, "Flow Event Telemetry on Programmable Data Plane," in *Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '20)*, pp. 76–89, ACM, 2020.

[21] C. Misa, W. O'Connor, R. Durairajan, R. Rejaie, and W. Willinger, "Dynamic Scheduling of Approximate Telemetry Queries," in *Proceedings of the 19th USENIX Symposium on Networked Systems Design and Implementation (NSDI '22)*, pp. 701–717, USENIX Association, 2022.

[22] G. Vassoler, J. A. Marques, and L. P. Gaspary, "VERMONT: Towards an In-band Telemetry-Based Approach for Live Network Property Verification," in *Proceedings of the 2023 IEEE/IFIP Network Operations and Management Symposium (NOMS '23)*, IEEE, 2023.

[23] Z. Xu, Z. Lu, and Z. Zhu, "Information-Sensitive In-Band Network Telemetry in P4-Based Programmable Data Plane," *IEEE/ACM Transactions on Networking*, 2024.

[24] K. Papadopoulos, P. Papadimitriou, and C. Papagianni, "Deterministic and Probabilistic P4-Enabled Lightweight In-Band Network Telemetry," *IEEE Transactions on Network and Service Management*, vol. 20, no. 4, pp. 4909–4922, 2023.

[25] H. N. Nguyen, B. Mathieu, M. Letourneau, G. Doyen, S. Tuffin, and E. M. d. Oca, "A comprehensive p4-based monitoring framework for l4s leveraging in-band network telemetry," in *NOMS 2023-2023 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–6, 2023.

[26] Z. Liu, A. Manousis, G. Vorsanger, V. Sekar, and V. Braverman, "One Sketch to Rule Them All: Rethinking Network Flow Monitoring with UnivMon," in *Proceedings of the 2016 Conference of the ACM Special Interest Group on Data Communication (SIGCOMM '16)*, pp. 101–114, ACM, 2016.

[27] O. Michel, J. Sonchack, G. Cusack, M. Nazari, E. Keller, and J. M. Smith, "Software Packet-Level Network Analytics at Cloud Scale," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 597–610, 2021.

[28] R. Doriguzzi-Corin, L. A. D. Knob, L. Mendozzi, D. Siracusa, and M. Savi, "Introducing packet-level analysis in programmable data planes to advance network intrusion detection," *Computer Networks*, vol. 239, p. 110162, 2024.

[29] Y. Hu, D.-M. Chiu, and J. C. S. Lui, "Adaptive Flow Aggregation - A New Solution for Robust Flow Monitoring under Security Attacks," in *Proceedings of the 2006 IEEE/IFIP Network Operations and Management Symposium (NOMS '06)*, pp. 424–435, IEEE, 2006.

[30] B. Guan and S.-H. Shen, "FlowSpy: An Efficient Network Monitoring Framework Using P4 in Software-Defined Networks," in *Proceedings of the 2019 IEEE 90th Vehicular Technology Conference (VTC '19-Fall)*, IEEE, 2019.

[31] J. Sonchack, A. J. Aviv, E. Keller, and J. M. Smith, "TurboFlow: Information Rich Flow Record Generation on Commodity Switches," in *Proceedings of the 13th European Conference on Computer Systems (EuroSys '18)*, ACM, 2018.

[32] F. Erlacher, W. Estgfaeller, and F. Dressler, "Improving Network Monitoring through Aggregation of HTTP/1.1 Dialogs in IPFIX," in *Proceedings of the 2016 IEEE 41st Conference on Local Computer Networks (LCN '16)*, pp. 543–546, IEEE, 2016.

[33] N. Gray, K. Dietz, M. Seufert, and T. Hossfeld, "High Performance Network Metadata Extraction Using P4 for ML-based Intrusion Detection Systems," in *Proceedings of the 2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR '21)*, IEEE, 2021.

[34] D. Barradas, N. Santos, L. Rodrigues, S. Signorello, F. M. V. Ramos, and A. Madeira, "FlowLens: Enabling Efficient Flow Classification for ML-based Network Security Applications," in *Proceedings of the 28th Annual Network and Distributed System Security Symposium (NDSS '21)*, Internet Society, 2021.

[35] G. Gori, L. Rinieri, A. Al Sadi, A. Melis, F. Callegati, and M. Prandini, "GRAPH4: A Security Monitoring Architecture Based on Data Plane Anomaly Detection Metrics Calculated over Attack Graphs," *Future Internet*, vol. 15, no. 11, 2023.

[36] P. Velan, "EventFlow: Network flow aggregation based on user actions," in *Proceedings of the 2016 IEEE/IFIP Network Operations and Management Symposium (NOMS '16)*, pp. 767–771, IEEE, 2016.

[37] P. Haag, "nfdump." https://github.com/phaag/nfdump, 2015.

[38] I. B. Fink, I. Kunze, P. Hein, J. Pennekamp, B. Standaert, K. Wehrle, and J. Rüth, "Prototype Implementation of HybridMon." https://github.com/COMSYS/HybridMon, 2025.

[39] L. F. Sikos, "Packet analysis for network forensics: A comprehensive survey," *Forensic Science International: Digital Investigation*, vol. 32, 2020.

[40] B. Claise, "Cisco Systems NetFlow Services Export Version 9." RFC 3954, 2004.

[41] B. Claise, B. Trammell, and P. Aitken, "Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information." RFC 7011, 2013.

[42] C. Gates, M. Collins, M. Duggan, A. Kompanek, and M. Thomas, "More Netflow Tools for Performance and Security," in *Proceedings of the 2004 USENIX Large Installation System Administration Conference (LISA '04)*, pp. 121–132, USENIX Association, 2004.

[43] nTop, "nProbe." https://www.ntop.org/products/netflow/nprobe/, 2015.

[44] T. AbuHmed, A. Mohaisen, and D. Nyang, "Deep Packet Inspection for Intrusion Detection Systems: A Survey," *Journal of the Korean Institute of Communication Sciences*, vol. 24, no. 11, pp. 22–36, 2007.

[45] A. Kuzmanovic and E. W. Knightly, "Low-rate TCP-targeted denial of service attacks: the shrew vs. the mice and elephants," in *Proceedings of the ACM SIGCOMM 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '03)*, pp. 75–86, ACM, 2003.

[46] J. R. Binkley and B. Massey, "Ourmon and Network Monitoring Performance," in *Proceedings of the FREENIX Track: 2005 USENIX Annual Technical Conference (ATC '05)*, pp. 95–108, USENIX Association, 2005.

[47] P. Velan and P. Čeleda, "Application-Aware Flow Monitoring," in *Proceedings of the 2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM '19)*, pp. 701–706, IEEE, 2019.

[48] J. Amann and R. Sommer, "Providing Dynamic Control to Passive Network Security Monitoring," in *Proceedings of the 18th International Symposium on Research in Attacks, Intrusions, and Defenses (RAID '15)*, vol. 9404, pp. 133–152, Springer, 2015.

[49] H. N. Nguyen, B. Mathieu, M. Letourneau, G. Doyen, S. Tuffin, and E. M. d. Oca, "A Comprehensive P4-based Monitoring Framework for L4S leveraging In-band Network Telemetry," in *Proceedings of the 2023 IEEE/IFIP Network Operations and Management Symposium (NOMS '23)*, IEEE, 2023.

[50] L. Castanheira, R. Parizotto, and A. E. Schaeffer-Filho, "FlowStalker: Comprehensive Traffic Flow Monitoring on the Data Plane using P4," in *Proceedings of the 2019 IEEE International Conference on Communications (ICC '19)*, IEEE, 2019.

[51] G. Gori, L. Rinieri, A. Al Sadi, A. Melis, F. Callegati, and M. Prandini, "Graph4: A security monitoring architecture based on data plane anomaly detection metrics calculated over attack graphs," *Future Internet*, vol. 15, no. 11, 2023.

[52] C. M. Inacio and B. Trammell, "YAF: Yet Another Flowmeter," in *Proceedings of the 2010 USENIX Large Installation System Administration Conference (LISA '10)*, USENIX Association, 2010.

[53] Intel Corporation, "Intel® Tofino™ Programmable Ethernet Switch ASIC." https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch/tofino-series.html, 2020.

[54] R. Ben-Basat, X. Chen, G. Einziger, and O. Rottenstreich, "Efficient Measurement on Programmable Switches Using Probabilistic Recirculation," in *Proceedings of the 2018 IEEE 26th International Conference on Network Protocols (ICNP '18)*, pp. 313–323, IEEE, 2018.

[55] IANA, "IP Flow Information Export (IPFIX) Entities." https://www.iana.org/assignments/ipfix/ipfix.xhtml, 2007.

[56] CAIDA, "The CAIDA UCSD Anonymized Internet Traces 2011 – DirA 20110607-235600 UTC." https://catalog.caida.org/details/dataset/passive_2011_pcap, 2011.

[57] CAIDA, "The CAIDA UCSD Anonymized Internet Traces 2011 – DirB 20151217-133400 UTC." https://catalog.caida.org/details/dataset/passive_2011_pcap, 2011.

[58] CAIDA, "The CAIDA UCSD Anonymized Internet Traces 2015 – DirA 20151217-133400 UTC." https://catalog.caida.org/details/dataset/passive_2015_pcap, 2015.

[59] CAIDA, "The CAIDA UCSD Anonymized Internet Traces 2018 – DirA 20180816-135200 UTC." https://catalog.caida.org/details/dataset/passive_2018_pcap, 2018.

[60] Progress Software Corporation, "Flowmon Probe." https://www.flowmon.com/en/products/appliances/probe, 2009.

[61] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection," in *Proceedings of the 25th Annual Network and Distributed System Security Symposium (NDSS '18)*, Internet Society, 2018.

[62] Y. Mirsky, "Kitsune Demo Code." https://github.com/ymirsky/Kitsune-py/blob/master/example.py, 2018.

[63] M. Dahlmanns, J. Lohmöller, I. B. Fink, J. Pennekamp, K. Wehrle, and M. Henze, "Easing the conscience with opc ua: An internet-wide study on insecure deployments," in *Proceedings of the ACM Internet Measurement Conference*, IMC '20, (New York, NY, USA), pp. 101–110, Association for Computing Machinery, 2020.

[64] H. Böck, "Vulnerabilities show why STARTTLS should be avoided if possible." https://blog.apnic.net/2021/11/18/vulnerabilities-show-why-starttls-should-be-avoided-if-possible/, 2021 (accessed October 11, 2024).