LANTERN: Layered Adaptive Network Telemetry Collection for Programmable Dataplanes

Kaiyu Hou[*]
Alibaba Cloud

Dhiraj Saharia
Georgetown University

Vinod Yegneswaran
SRI International

Phillip Porras
SRI International

ABSTRACT

Managing next-generation enterprise networks requires collecting and analyzing enormous volumes (tens of Tbps) of network traffic data in real time to detect potential anomalies, classify attacks, identify root causes, and rapidly deploy effective mitigations. Conducting robust and scalable analysis on such traffic volumes is a daunting "haystack" problem that demands intelligent strategies to winnow traffic to extract and pinpoint "needles" of interest. Recent advances in software-defined networking and programmable dataplanes, that enable dynamic reconfiguration of switching hardware to adapt to changing traffic conditions, provide a foundational building block. However, they lack the resources and programming primitives for complex computational models.

Toward that end, we present LANTERN, a layered and adaptive network telemetry system that facilitates joint collection and analysis of network traffic at multiple resolutions in coordination with the controller. Our design offloads complex machine-learning analysis to the controller, while still enabling *proactive* telemetry refinement and *reactive* mitigation triggers at the data-plane level. We evaluate our layered approach by replaying a labeled CIC-IDS attack dataset through both software and hardware P4 switches. LANTERN is able to detect most anomalies, accurately classify them, and introduces negligible switching overhead (1% latency).

CCS CONCEPTS

• **Networks → Programmable networks**; • **Computing methodologies →** *Anomaly detection.*

KEYWORDS

P4; Programming Switch; Anomaly Detection

ACM Reference Format:
Kaiyu Hou, Dhiraj Saharia, Vinod Yegneswaran, and Phillip Porras. 2023. LANTERN: Layered Adaptive Network Telemetry Collection for Programmable Dataplanes. In *Proceedings of the 6th European P4 Workshop (EuroP4 '23), December 8, 2023, Paris, France.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3630047.3630194

[*]This work was initiated while Kaiyu Hou was a research intern at SRI International.

1 INTRODUCTION

Emerging next generation networks offer many promises, such as dramatic bandwidth expansion (due to technologies like 5G), significantly lower latency (< 1 ms), and billions of connected devices. If realized, this new 5G reality will provide the foundations for (*a*) new *IoT everywhere* device communication models, (*b*) new trust models (driven by greater communication), and (*c*) new service delivery models. Unfortunately, this generational leap forward in networking will provide equivalent leaps in the capabilities of attackers.

An important requirement for detecting attacks directed against these next-generation networks is the ability to gather scalable and fine-grained network measurements from network devices. Recent breakthroughs in programmable routing silicon overcome the limitations of conventional router ASICs by leveraging massive parallel processing capacity to facilitate deep packet inspection capabilities while delivering terabit-level forwarding capacity. An important component is developing algorithmic techniques that leverage these hardware capabilities to enable scalable *always-on in-band* telemetry collection query frameworks for efficiently performing analytics on this data. Prior efforts such as Sonata [5], have attempted to address this problem by developing a framework that allows network operators to formulate telemetry questions using MapReduce-style queries across network traffic. They showed how to formulate queries that would allow operators to quickly and efficiently detect security events such as DNS reflection attacks and showed how these queries could be mapped to various network targets, particularly P4 switches.

We describe our approach in the context of a system called LANTERN that has three analysis layers, but that can be extended with additional intermediate layers. At the first layer, LANTERN explores variations in traffic characteristics at the link level. One approach that we explore is the use of variational autoencoder (VAE) [8] schemes that are designed to model the internal operations of the P4 switch itself, based on switch-internal attributes, interface statistics, errors, and event counts. The approach can deliver network stability insights in a highly telemetry-efficient manner, using features that operate within a sufficiently tractable bound of input dimensionality. This perspective, while not fine-grained for conducting malicious-flow or attack-source analyses, could provide a scalable technique for detecting catastrophic modality shifts in switch operations, which can then *trigger* finer-grained feature extractions that drive root-cause diagnoses.

Unexpected deviations in link-level characteristics trigger additional fine-grained statistics tracking at the flow-level, which corresponds to the second layer of LANTERN. Finally, subsets of deviant flows are subject to deep packet analysis by careful sampling of payload bytes that are then sent to the controller. The controller uses Rabin Fingerprints [11] to identify invariant content that is specific to offending flows and deploys a *mitigating trigger* at the P4 switch to automatically block all traffic from IP addresses that demonstrate the anomalous traffic fingerprint. Our guiding principle in designing this framework is to minimize real-time controller interaction such that the switch can proactively adapt based on periodic guidance from the controller. In addition, the ultimate objective is to enable the switch with capability to install dynamic mitigation rules that are informed both by the layered analysis and apriori triggers generated by the controller.

Contributions: The contributions of our work include:

- A novel framework for layered telemetry collection in programmable networks;
- Strategies for VAE-based attack detection and supervised attack classification techniques;
- Concept of dynamic P4 response triggers based on Rabin Fingerprints for proactive mitigation;
- Prototype implementation in BMV2 and Tofino ASIC and evaluation using the CIC-IDS dataset.

2 BACKGROUND AND MOTIVATION

We provide an overview of the types of network attacks and anomalies that we seek to detect and mitigate. Next, we describe the types of telemetry that we would need to isolate such phenomenon and highlight advantages of LANTERN over published and state-of-the-art approaches. The classes of anomalous network events of interest broadly include inbound and outbound DDoS attacks, large-scale network reconnaissance, worm outbreaks, flash crowds, and network outages.

Network Attacks. DDoS attacks have been well studied in the context of fixed enterprise and core networks. These include host-based resource-exhaustion attacks [3], application-level attacks [1], reflector-based amplification attacks [13], pulsing attacks [10], temporal lensing attacks [12], and link-flooding attacks [7]. For each of these attacks, key statistics to consider include volume of traffic, distribution of sources, and specific timing patterns.

Large-scale reconnaissance attempts that horizontally attempt to map the network produce unusually high amounts of failed connection attempts, without retries. Worm outbreaks typically manifest themselves at the network layer as upticks in the number of flows and sources directed at specific ports, but not specifically directed at any particular host. In contrast, flash crowds resemble more like a DDoS attack in that they demonstrate sudden spikes in flow counts and IP source counts connecting to a specific service. The key difference is that in a well-engineered, elastic network this does not result in measurable degradation in service response times. Finally, network outages are characterized by unusual delays, reconnection attempts, and retransmits.

Network security monitoring and situational awareness in response to such threats require collecting network statistics and metrics in real time. Recent breakthroughs in network telemetry lie in the field of "reactive" collection. They require the network operators to issue series of commands to the dataplane to query multiple statistics or a combination of several metrics to detect and isolate traffic anomalies. The decision process involved in making a mitigation decision typically involves answering some of the following questions: (*i*) Is my network experiencing an anomaly? (*ii*) Is this anomaly an attack or an outage? (*iii*) If an attack, is it an internal→external or external→internal attack? (*iv*) What type of attack is it? (*v*) Which systems are victims and sources of the attack? (*vi*) How do we mitigate this attack?

The objective of LANTERN is to enable the controller to dynamically program the switch with proactive functions that initiate fine-grained telemetry when the switch finds the network to be under duress or observes symptoms of anomalous behavior from always-on coarse-grained telemetry.

Programmable Data Planes for Threat Mitigation. Prior work has explored strategies for dynamic telemetry collection in programmable dataplanes. Notable such efforts include systems like Sonata, QPipe, Poseidon, Jaqen, Dynatos, and Stats 101, as summarized in Table 1. Sonata [5] provides an interface to express queries for some common telemetry tasks, such as querying newly opened TCP connections and identifying DNS reflection attacks. To achieve real-time execution, Sonata dynamically adjusts queries to ensure the availability of switch resources, such as fast memory. Here, the focus is on scalably supporting query traffic, while the network operators are agnostic to where or how the query will execute. DynATOS [2] further improves the performance of Sonata by using a resource scheduler to dynamical decide when and for how long to execute the set of active queries on the data plane at runtime. It can also execute multiple concurrent and sequential queries on switches to meet the accuracy and latency goals. Poseidon [17] and Jaqen [9] propose modular policy abstractions and defense primitives to collect flow statistics on P4 switches and integrate DDoS detection with defense. LANTERN leverages programmability to support both reactive telemetry collection and proactive mitigation. Without manual interference, our learning engine can automatically reprogram the running switches to adaptively collect fine-grained telemetry as well as implement mitigating triggers that automatically block source IPs or subnets based on specific behavioral patterns.

2.1 Motivating Scenarios

We discuss two motivating scenarios. First, we consider the case of a new IoT vulnerability that is being actively exploited to launch DDoS attacks. Next, we describe how LANTERN can detect and mitigate an outbound data exfiltration attack.

Scenario 1: Internal->External IoT DDoS attack. An anomaly may be observed in internal scanning as well as outbound flows and/or overall traffic. Many internal hosts may suddenly start exhibiting this behavior.

- *Step 1:* Data plane stats are used by the controller to identify that there is an anomaly.
- *Step 2:* Control plane identifies the type of attack and its direction.
- *Step 3:* Control plane isolates the networks, ports, or sources involved in the attack
- *Step 4:* Control plane instructs data plane to employ selective DPI to collect payload from associated attack sources.
- *Step 5:* Control plane identifies this is a new attack and uses Rabin fingerprints to generate payload/flow-level signature for it.

Table 1: Design Space of Published Telemetry Collection Methods for Programmable Networks

Collection Methods	Approach	Controller Involved	Adaptive Data Plane Telemetry	Data Plane Alerting	Mitigation Triggers	Resources Optimized
LANTERN	Layer-Based ML	Yes	Layer-based	Yes	Diverse attacks	Memory, controller-bandwidth, match entries
Stats 101 [4]	Online Computation	Yes	Yes by Controller	Yes	No	High DP Memory and Processor Usage
QPipe [6]	Sketch-Based	No	No	No	No	Low Control Bandwidth Usage due to Sketches
Sonata [5]	Reactive Query	Yes	Reactivate by Admin	No	By Admin	Reduce DP Processor Usage by workload ordering
DynATOS [2]	Reactive Query	Yes	Reactivate by Admin	No	By Admin	Reduce DP Query Overhead by runtime ordering
Poseidon [17]	Predefined DP Policy	Yes	Policy-based	Yes	DDoS only	Low DP Memory; Required 3rd-party scrubbing centers
Jaqen [9]	Sketch-based	Yes	No	Yes	DDoS only	Match entries, hash bits, SRAM

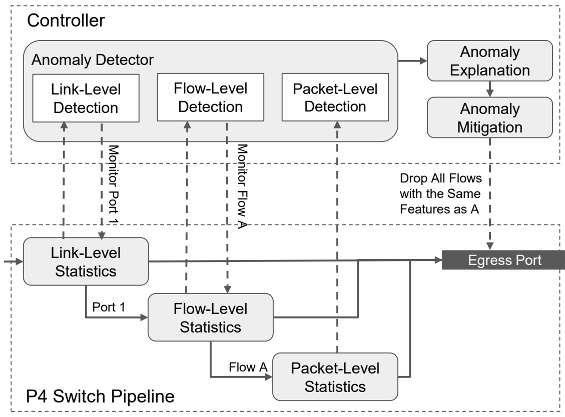


Figure 1: System design overview of LANTERN.

- *Step 6:* Control plane deploys proactive mitigation triggers at the switch such that all flows from new sources exhibiting this anomalous behavior are automatically blocked at the switch without involving the controller.

Scenario 2: Internal-External Data Exfiltration. An anomalous internal to external communication may involve exfiltration of sensitive data to an untrusted remote network.

- *Step 1:* Coarse-grained LANTERN data plane statistics are used by the controller to identify that there is an anomalous high volume data transfer from one or more internal hosts to a remote host.
- *Step 2:* LANTERN control plane queries the data plane for more fine-grained statistics on precursor inbound flows to the victim IP address.
- *Step 3:* If necessary, the LANTERN control plane installs triggers to obtain deeper statistics of flows communicating with the suspected external network.
- *Step 4:* Once the attack has been accurately characterized, mitigating triggers are deployed to block future exfiltration attacks that demonstrate similar communication patterns.

3 SYSTEM DESIGN

Figure 1 provides a high-level illustration of the design of LANTERN and how it enables layered telemetry collection for adaptive threat mitigation. It is composed of two major parts: (*i*) an execution engine at P4 programmable data plane, and (*ii*) an analysis engine at the controller side. The P4 data plane is primarily for online telemetry collection and can also mitigate threats by executing new flow rules. The controller can perform more advanced analysis based on the telemetry collected at dataplane and can also install new flow rules to the data plane. In addition, it can install triggers that instruct the switch to proactively deploy flow rules when certain network conditions are met.

The layer-based architecture of LANTERN allows the system to balance attack detection fidelity while optimizing for data plane memory/computation usage and control channel bandwidth utilization. Specifically, the current design incorporates three levels of telemetry collection and analysis modules in both the data plane and the controller, namely *link-level*, *flow-level*, and *packet-level*.

At the link-level collection, the P4 switch will only collect very coarse-grained telemetry on all the traffic at each ingress port. Specifically, it will only collect seven online port statistics. The controller uses these statistics for anomaly detection. We have implemented a simple unsupervised VAE analysis based on these statistics. Our motivation behind implementing VAE-based algorithms is that they involve simple matrix multiplication tasks that can be natively ported to the switch in the future. Our preliminary evaluation shows that these seven feature statistics are already able to assist in detecting anomalies and triggering alerts (though with limited false positives).

When an anomaly is detected by the VAE algorithm through the layer-1 module, the telemetry collection will then proceed to activate flow-level collection. At this level, we collect more fine-grained telemetry for each flow on the targeted switch ingress port, based on features suggested by NetML [16]. We describe how these are collected with greater detail in Section 4. The controller may then run more fine-grained attack classification algorithms based on flow-level telemetry. Specifically, we have implemented a supervised N-way classification algorithm. The detection algorithm will try to identify flows that trigger the anomaly and also identify the kinds of attacks that may correspond to this anomaly.

The third layer for telemetry collection is at the packet level. After identifying the attack type from layer-2 analysis, the task for layer-3 is to produce attack signatures and generate mitigation flow rules based on packet sampling. Specifically, the P4 data plane will sample packets from the targeted flow and send them to the controller. The controller will generate signatures, using invariant sequence analysis techniques such as Rabin Fingerprints, and produce flow rules or proactive triggers based on such signatures for attack mitigation.

3.1 Layer-based P4 Telemetry Collection

Programmability is a fundamental advantage of the P4 data plane. Yet, compared to general-purpose programming languages, this ability is highly-constrained within the P4 environment. For instance, there are no loop instructions or division operations in the

P4 language, there is only limited information we can extract from a packet (type, length, and value) and the packet processing in P4 is largely stateless, i.e., only counters and registers, which have very limited capacity, can be used to store stateful information.

First, we briefly describe the methods we used to collect traffic statistics for different levels in the P4 data plane and discuss the main challenges. Our statistics collection methods draw from published research ([4, 6]) that has implemented complex mathematical operations (e.g., square root, median estimation, and quantiles) and data structures using P4.

Flow Identification. We implement a Bloom filter to track switch-level statistics, including `new_flows`, `active_flows`, and `finished_flows`. We use the built-in P4 hash function to implement a simple Bloom filter with 2 bits. When a new SYN packet is observed, we increase the `new flows` and `active flows` counter. When a FIN packet is observed, we check if it has a corresponding SYN packet and update the `finished_flow` and `active_flow` counters.

Flow ID. It is challenging to use the limited switch space to map a flow to its corresponding 5-tuple (src IP, dest IP, src port, dest port, and protocol). For each new flow, we use a hash function to map its 5-tuple to a `flow_id` value. All telemetry corresponding to this flow is associated to this `flow_id` value. We also use 4 registers to store the mapping between a `flow_id` to its corresponding src IP, dest IP, src port, and dest port. (In our preliminary implementation, we only consider TCP flows. Therefore, we do not need to record the protocol in the 5-tuple). With these registers, after detecting a flow as anomaly, the controller can generate mitigation rules by mapping the `flow_id` back to its corresponding 5-tuple.

Average Flow Duration. We estimate the average flow duration from the P4 data plane as shown in Code 1. Other telemetry statistics are collected similarly. The flow start time is recorded to a register and indexed by the `flow_id` value (L3). The flow duration is calculated when the flow is finished (L8). Since P4 does not support the divide operation and fractions, we use the formula $ave\_duration = (ave\_duration * 15 + flow\_duration) >> 4$ with bit shifting to estimate the recent average flow duration.

3.2 Layer-based Decision Engine in Controller

Link-level Telemetry for Anomaly Detection. To enable link-level anomaly detection, we collect coarse-grained telemetry on each switch link (switch port) and use VAEs [8] to detect network anomalies. VAEs are attractive for anomaly detection as they are unsupervised detection techniques that can be trained online with unlabeled datasets. We consider network anomaly events to be points where the evaluation loss is higher than the average training loss by a multiple (α) of the standard deviation. We prefer a low threshold to trigger the alert as a false positive alert can be corrected later in the flow-level attack classification with more fine-grained telemetry. We collect seven statistics in the P4 dataplane for VAE training and evaluation, including number of new flows, finished flows, active flows in five seconds, number of active flows in 60 seconds, average flow duration, total packets per second, and total bytes per second.

Flow-level Telemetry for Attack Classification. We consider flow-level detection as a multi-class classification problem. In this layer, the classification model is pre-trained by labeled traffic. Taking the

```
1 // tcp.syn
2 cur_time = standard_metadata.
      ingress_global_timestamp;
3 flow_time_counter.write(flow_id, cur_time);
4 // tcp.fin
5 cur_time = standard_metadata.
      ingress_global_timestamp;
6 flow_time_counter.read(last_time, flow_id);
7 flow_duration = (cur_time - last_time);
8 ave_time = (ave_time * 15 + flow_duration) >> 4;
```

Code 1: Average Flow Duration Estimation

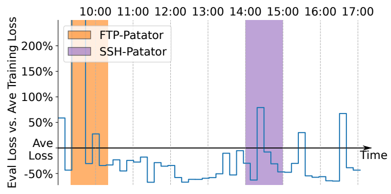
telemetry of each flow as the input, we aim to classify whether the given flow is benign or attack, and determine the attack type. For the prototype verification, we use a simple decision tree model as the classifier. It can already generate classification accuracy of > 98% for most attacks that we evaluated. We expect a more advanced multi-class classifier can achieve better results. Based on a previous study [16], we collect statistics of number of packets, number of bytes, average packet size, minimal packet size, maximal packet size, size of the first ten packets, and inter-arrival time of the first ten packets for each flow.

Packet-level Telemetry for Attack Mitigation. Our next goal is to use more fine-grained packet-level telemetry to mitigate similar inbound attack flows, which could come from other source addresses than ones we have already observed and blocked. After the controller classifies an attack flow, it instructs the switch to sample packets from the flow to create dynamic mitigation rules. Depending on the attack type, the mitigation rule could be as simple as targeting the 5-tuple. It could also target a /24 subset with more fine-grained mitigation rules, such as dropping flows with the same size and inter-arrival time distribution of the first ten packets. Moreover, attacks can also be mitigated based on packet signatures. Since we do not want to affect the packet processing rate at the data plane, we consider the attack mitigation as two tasks: (*i*) the attack signature generation task in the controller and (*ii*) the signature matching task at the P4 data plane. Specifically, we use the controller to generate the attack signatures. Those signatures are based on sampled packets from the attack flow which were identified in the second step.

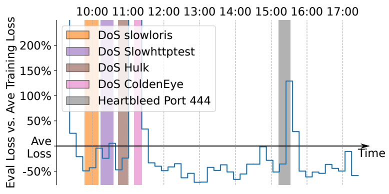
The controller then proceeds to install attack mitigation rules at the P4 data plane. Any incoming flows, which contain the attack signatures, will be automatically blocked by the P4 data plane. After the mitigation rules are installed, the mitigation process does not require any assistance from the controller. Therefore, the signature matching process could be done very quickly in the P4 hardware.

4 SYSTEM IMPLEMENTATION

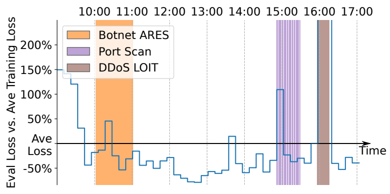
We discuss the challenges involved in porting the BMV2 implementation to Tofino. There are many ways in which the real hardware implementation differed from the BMV2 implementation. (*i*) The BMV2 v1model architecture has no limitation on addition and multiplication of signed and unsigned integer types. (*ii*) There is no limitation on the bit-widths of the types of register arrays and counters that we can instantiate. (*iii*) The metadata attributes such as packet length are defined at all phases of the packet processing



(a) Tuesday



(b) Wednesday



(c) Friday

Figure 2: VAE-based link-level telemetry results. Model is trained on Monday's telemetry (benign traffic). The colored ranges are when the attack happened. Alerts are generated for all attacks based on the link-level telemetry except slow DoS.

Algorithm 1: Fingerprint-based attack mitigation

```
Input  : $attack\_flows, benign\_flows$
Output : $mitigation\_rules$
1 for flow in attack_flows do
2     $cand\_sigs \leftarrow cand\_sigs \cap \{$Rabin Fingerprint$(flow[1:4])\}$
3 for flow in background_flows do
4     $cand\_sigs \leftarrow cand\_sigs - \{$Rabin Fingerprint$(flow[1:4])\}$
5 for sig in cand_sigs do
6     $rules \leftarrow rules \cup \{$Match:$raw(sig)$, Action:$reject(src\_ip)\}$
```

pipeline. (*iv*) There is no limitation on reading and writing a register multiple times per packet processed. (*v*) There is no limitation on the arithmetic that can be done to "update" values inside of a register array. (*vi*) There is no limitation on the size of bit-shift operations. (*vii*) There is no limitation on the bit-widths of integer types that we wish to compare.

When we tried to perform a direct translation of the Bloom filter code in the v1model, we were unable to due to limitations not just on the number of stages, but also on the arguments having to be aligned in certain locations in memory before they were passed to another phase of processing. For this reason, we split the number of actions from 1 compute_hashes action, to 3 of them (1 for each hash we needed to calculate). We then directly store that value into the ingress bridged header so that we can resume computation on these fields in the egress processing.

Fingerprint generation and attack mitigation. Based on prior work on automated worm fingerprinting [15], we use Rabin fingerprinting [11] to generate Layer-7 packet signatures from the attack flows in four steps. First, we only sample the first N (we use N=4) packets from the attack flows. Second, we only consider fingerprints with high content prevalence among the flows from this attack as the candidate attack fingerprints. Third, we consider all fingerprints shown in background traffic as benign signatures. Hence, we generate signatures from non-attack periods and eliminate them from the candidate attack fingerprints. Finally, we map attack fingerprints back to the raw bytes strings and use these strings as the matching keys of the mitigation rules, as described in Algorithm 1. The P4 switch will block source IPs that triggered these newly installed mitigation rules.

5 SYSTEM EVALUATION

Dataset. We evaluate LANTERN using the CIC-IDS intrusion detection evaluation dataset [14]. This dataset has been widely used in recent studies for evaluating network intrusion detection and anomaly detection algorithms ([2, 16]). It has raw PCAP captures with ~50 million packets (50 GB) in five days, labeled from Monday to Friday. The dataset contains both benign and the most common attacks, including botnet, port scan, DDoS, Patator, and Heartbleed port 444. The Monday traffic only includes the benign traffic. For the following days, attacks are labeled by the combination of types, time ranges, attacker and victim IPs. Therefore, we can directly identify the flows corresponding to an attack event. This information is used to verify the efficiency of our algorithms.

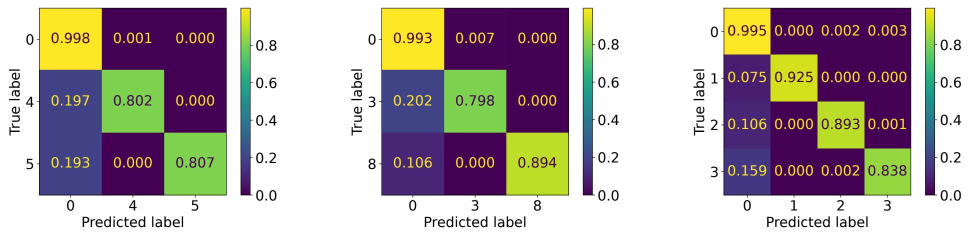
Testbed. We use two end hosts, one BMV2 software P4 (or hardware Tofino) switch, and one controller in our evaluation topology. The two end hosts connect each other through the P4 switch. One host is sending the raw PCAP of CIC-IDS dataset at the original capturing rate by using the TCPReplay command. The P4 switch forwards every packet it received from one port to another port. While forwarding traffic, the P4 switch also runs our layer-based adaptive telemetry collection algorithm and synchronizes statistics to the controller. The controller dynamic adjusts the telemetry collection policy and installs new mitigation rules in the switch.

5.1 Link-Level Detection

Figure 2 shows the results of applying VAE to the link-level telemetry of (a) Tuesday, (b) Wednesday, and (c) Friday. The purpose of link-level detection is to determine whether there is an anomaly happening in a specific switch link (*i.e.*, switch port) with very limited telemetry, and dive into the flow-level detection for all flows in that switch link. The VAE model is trained using the link-level telemetry collected on Monday's traffic. The x-axis in these figures represents the time in one day from 9:00 am to 17:00 pm. The colored ranges are when attacks occur. The y-axis represents the loss function value difference. We simply compare the evaluation traffic loss against the average training loss of Monday's traffic. Even when we use simple detection thresholding, after the initial stage, the VAE detector can trigger alerts for the vast majority of attack events (except slow DoS). Additionally, there are multiple false-positive alerts which can be winnowed using flow-level detection.

5.2 Flow-Level Detection

Figure 3 shows the results of applying the basic decision tree classifier to the flow-level telemetry of (a) Tuesday, (b) Wednesday, and (c) Friday. The purpose of flow-level detection is to determine the attack type of an anomaly event. Unlike the link-level detection, the VAE model is trained by unsupervised data. At this level, we use 80% of the labeled attack traffic to train the decision tree, and use the rest 20% of traffic to evaluate the model. Specifically, we consider all



(a) Tuesday (b) Wednesday (c) Friday

Figure 3: Results of applying basic decision tree to the flow-level telemetry for the Tofino testbed. We randomly split 80% data for training and 20% data for verification. The flow-level telemetry can be used to identify the type of attacks with high accuracy.

```
'\nAccept-Encoding: gzip, deflate\r\nAccept'
'on: keep-alive\r\nAccept-Encoding: gzip,'
'08.1 HTTP/1.1\r\nHost: 205.174.165.73:808'
```

Code 2: Exemplar Benign Signatures

```
't: */*\r\nUser-Agent: python-requests/2.1'
': */*\r\nUser-Agent: python-requests/2.14'
'*/*\r\nUser-Agent: python-requests/2.14.2'
...
'api/pop?botid=mitacs-pc6&sysinfo=Window'
'api/pop?botid=mitacs-pc4&sysinfo=Window'
...
'botid=mitacs-pc6&sysinfo=Windows%207 HT'
'botid=mitacs-pc4&sysinfo=Windows%2010 H'
..
```

Code 3: Exemplar Attack Signatures

DoS attacks to be one attack type. From the confusion matrix, we see that the decision tree can classify the benign traffic and types of attacks with very high accuracy. In addition, the training data is also highly unbalanced. For instance, there are more than 1 million data points labeled benign, while only 9000 data points are labeled as the FTP-Patator attack.

5.3 Packet-Level Attack Mitigation

To verify the fingerprint generation and mitigation algorithm, we use our framework to identify attack flows in Friday's data from 10 am to 11 am using the process as described in §4. Specifically, we consider all attack flows from the first fifteen minutes of traffic (10:00 - 10:15) that were identified as attack flows by flow-level telemetry. We sampled all fingerprints in the direction from attackers to the victim and only labeled 346 fingerprints which are observed in more than half of attack flows as candidate attack fingerprints.

Next, we consider all fingerprints shown in Monday's traffic as benign signatures and filter them out from the candidate attack fingerprints, to winnow down to a total of 235 fingerprints. Code 2 shows three typical benign signatures which have been filtered out from the candidate set. Keywords shown in these three signatures, including `gzip, deflate` (L1), `keep-alive` (L2), and `Host` with the victim's IP (L3), could be observed in every connection to the victim. So, it is reasonable that they are identified as benign signatures.

We apply all 235 attack fingerprints to Friday's traffic from 10:15 am to 11:00 am. There are in total of 27,225 flows. Our detection algorithm captured all 187 attack flows with zero false positives and zero false negatives. Code 3 shows some exemplar attack signatures.

5.4 Resource Usage and Performance

Hardware utilization. We measured the Tofino data plane's resource usage when LANTERN is deployed. The switch contains 12 stages and 6 stages were used for table allocation and the total number of tables allocated were 35. LANTERN does not use TCAM, as no ternary matching is performed. It uses an average of 8.75% total SRAM for exact matches, along with 6.77% VLIW (Very Long Instruction Word) entries, and 13.89% Map RAM across all the stages.
Latency. To measure the latency incurred by the additional processing of LANTERN, we compared the RTT of LANTERN against a basic port forwarding program for the Tofino testbed. We found that the latency of LANTERN is 613 ns, whereas 606 ns for the basic program, which means LANTERN adds a negligible latency of 7 ns on average across 2000 packets.

6 CONCLUSION

We have described the design and implementation of LANTERN, a framework for layered, adaptive telemetry collection in programmable dataplanes. The system includes capabilities to dynamically collect telemetry at multiple resolutions and deploy proactive mitigating triggers to block emerging threats. We have validated the system using both software switch (BMV2) and hardware (Tofino)-based implementations on the CIC-IDS dataset. Our results indicate that LANTERN is able to accurately detect a broad range of anomalies, characterize them, and deploy effective mitigating triggers while introducing minimal resource overhead and processing latency at the switch. Deeper exploration using additional datasets and comprehensive scalability testing is future work.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable feedback. We also thank Shivam Patel for his assistance in porting the original BMV2 implementation to Tofino. This work was partially supported by the NSF under grants CNS-2229455 and CNS-1935966. Any opinions, recommendations, or findings are those of the authors and do not reflect the views of NSF and Alibaba Cloud.

REFERENCES

[1] Enrico Cambiaso, Gianluca Papaleo, Giovanni Chiola, and Maurizio Aiello. 2013. Slow DoS attacks: definition and categorisation. *International Journal of Trust Management in Computing and Communications* 1, 3-4 (2013), 300–319.

[2] Misa Chris, OĆonnor Walt, Durairajan Ramakrishnan, Rejaie Reza, and Willinger Walter. 2022. Dynamic Scheduling of Approximate Telemetry Queries. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*.

[3] Scott A Crosby and Dan S Wallach. 2003. Denial of Service via Algorithmic Complexity Attacks.. In *USENIX Security Symposium*. 29–44.

[4] Sam Gao, Mark Handley, and Stefano Vissicchio. 2021. Stats 101 in P4: Towards In-Switch Anomaly Detection. In *Proceedings of the Twentieth ACM Workshop on Hot Topics in Networks*. 84–90.

[5] Arpit Gupta, Rob Harrison, Marco Canini, Nick Feamster, Jennifer Rexford, and Walter Willinger. 2018. Sonata: Query-driven streaming network telemetry. In *Proceedings of ACM SIGCOMM*. 357–371.

[6] Nikita Ivkin, Zhuolong Yu, Vladimir Braverman, and Xin Jin. 2019. Qpipe: Quantiles sketch fully in the data plane. In *Proceedings of the 15th International Conference on Emerging Networking Experiments And Technologies*. 285–291.

[7] Min Suk Kang, Soo Bum Lee, and Virgil D Gligor. 2013. The crossfire attack. In *2013 IEEE symposium on security and privacy*. IEEE, 127–141.

[8] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[9] Zaoxing Liu, Hun Namkung, Georgios Nikolaidis, Jeongkeun Lee, Changhoon Kim, Xin Jin, Vladimir Braverman, Minlan Yu, and Vyas Sekar. 2021. Jaqen: A High-Performance Switch-Native Approach for Detecting and Mitigating Volumetric DDoS Attacks with Programmable Switches. In *30th USENIX Security Symposium*.

[10] Xiapu Luo and Rocky KC Chang. 2005. On a new class of pulsing denial-of-service attacks and the defense.. In *NDSS*.

[11] Michael O Rabin. 1981. Fingerprinting by random polynomials. *Technical report* (1981).

[12] Ryan Rasti, Mukul Murthy, Nicholas Weaver, and Vern Paxson. 2015. Temporal lensing and its application in pulsing denial-of-service attacks. In *2015 IEEE Symposium on Security and Privacy*. IEEE, 187–198.

[13] Christian Rossow. 2014. Amplification Hell: Revisiting Network Protocols for DDoS Abuse.. In *NDSS*. 1–15.

[14] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A Ghorbani. 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp* 1 (2018), 108–116.

[15] Sumeet Singh, Cristian Estan, George Varghese, and Stefan Savage. 2004. Automated Worm Fingerprinting. In *Proc. 6th USENIX OSDI*. San Francisco, CA.

[16] Kun Yang, Samory Kpotufe, and Nick Feamster. 2020. Feature Extraction for Novelty Detection in Network Traffic. *arXiv preprint arXiv:2006.16993* (2020).

[17] Menghao Zhang, Guanyu Li, Shicheng Wang, Chang Liu, Ang Chen, Hongxin Hu, Guofei Gu, Qianqian Li, Mingwei Xu, and Jianping Wu. 2020. Poseidon: Mitigating volumetric ddos attacks with programmable switches. In *the 27th Network and Distributed System Security Symposium (NDSS)*.