Poster: A Fast Monitor for Slow Network Attacks

Cuidi Wei
George Washington University

Shaoyu Tu
University of California Riverside

Toru Hasegawa
Shimane University

Yuki Koizumi
Osaka University

K. K. Ramakrishnan
University of California Riverside

Junji Takemasa
Osaka University

Timothy Wood
George Washington University

***Abstract*—Recent work has demonstrated how programmable switches can effectively detect attack traffic, such as denial-of-service attacks in the midst of high-volume network traffic. However, these techniques primarily rely on sampling- or sketch-based data structures that can only be used to approximate the characteristics of dominant flows in the network. As a result, such techniques are unable to effectively detect slow attacks such as SYN port scans, SSH brute forcing, or HTTP connection exploits, which do so by stealthily adding only a few packets to the network.**

In this work we explore how the combination of programmable switches, Smart network interface cards (sNICs), and hosts can enable fine-grained analysis of every flow in a cloud network, even those with only a small number of packets. We focus on analyzing packets at the start of each flow, as those packets often can help indicate whether a flow is benign or suspicious, e.g., by detecting an attack which fails to complete the TCP handshake in order to waste server connection resources. Our approach leverages the high-speed processing of a programmable switch while overcoming its primary limitation – very limited memory capacity – by judiciously sending some state for processing to the sNIC or the host which typically has more memory, but lower bandwidth. Achieving this requires careful design of data structures on the switch, such as a bloom filter and flow logs, and communication protocols between the switch, sNIC, and host, to coordinate state.

***Index Terms*—Traffic monitor, slow network attacks, programmable switches, smartNIC**

I. Introduction

To understand and counter threats, cloud operators deploy infrastructure to monitor systems and network traffic, to detect anomalies, and identify specific attacks in a timely manner. The infrastructure may directly protect against some attacks or provide alerts that trigger intervention, either automated or by an operator, to block or ameliorate the impact of those attacks. Designing a cost-efficient traffic monitoring and analysis infrastructure that can detect a range of network attacks within a high-rate traffic stream can be very valuable. Implementing a robust security monitor is crucial for both wide area networks and cloud environments to defend against diverse and sophisticated network attacks, ensuring the integrity and reliability of hosted services. Programmable data planes and monitoring environments built out of a variety of programmable devices, including P4-capable programmable switches (P4switches), smart network interface cards (sNICs, that may also be P4-capable), and hosts can be exploited to achieve this capability to find the proverbial 'needle-in-the-haystack'.

Programmable networking technologies developed in the last few years are adding tremendous power to the network data plane. However, they have varying capabilities. Programmable switches are capable of much higher data plane throughput, while programmable sNICs have more computing capability and memory capacity. Programmable switches are increasingly being suggested for monitoring, using queries processed at Terabit link rates [1], [2]. At the same time, sNICs also increasingly support programmability beyond simple protocol offloading. Switches and sNICs both provide programmability, with a malleable data path driven by the capabilities of the P4 programming language. sNICs enable end-hosts to handle to fairly high-speed traffic, at 40–100 Gbps, rates [3], [4], although not quite at a Terabit scale. Since sNICs support more general computations [5] than switches, it is very beneficial to use them for stateful packet processing, to complement the coarse-grained query processing of programmable switches. In this project we seek to combine 1) the scalability of programmable switches, 2) the reasonably high-speed packet processing, programmability, and significant amounts of computational capability and state (compared to switches) of sNICs, and 3) the flexible processing capabilities and storage of host-based CPU processing.

This combination of approaches is critical to support the large variety of attacks that target modern communication networks. There is a need to detect not only relatively crude volumetric attacks that overwhelm the network through sustained network activity (e.g., denial of service), but also more sophisticated attacks that probe for system weaknesses (*e.g.* SSH Brute forcing, port scan [6]), or attacks that exploit protocol dynamics (*e.g.* low-rate TCP attacks [7]).

Prior work has demonstrated how sketch data structures can be used to approximate flow-level statistics [8]. However, these approaches are only capable of monitoring the heaviest flows in the network. The data structures they use, such as NitroSketch [9], are focused on detecting heavy hitters, operating within the limits of the available amount of memory. However, their ability to detect slow attacks that send only a few packets is quite limited. Other works seek to complement programmable switches that count packets from a class of flows (e.g., a range of IP addresses) with sNICs and host that receive only a subset of traffic considered suspicious for

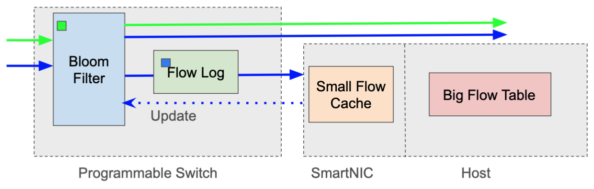

Fig. 1. Architecture of Monitoring Model

further detailed analysis [10]. However, slow attacks may still evade detection by [10], as attack flows may not be classified as suspicious when they comprise a relatively small number of packets.

In this poster we describe our ongoing work that seeks to harness heterogeneous data plane devices in order to build a network monitoring system that can efficiently and accurately observe *all* flows in the network. Similar to previous work, we focus on out-of-band monitoring, i.e., monitoring that is performed on a mirrored stream of traffic which does not incur additional latency or impose bandwidth constraints on the real traffic. Nevertheless, our monitoring system must be carefully designed to ensure it can keep up with traffic rates of 100s of gigabits or Terabits per second.

Our primary contributions include:

1. Leveraging programmable network devices for stateful monitoring of traffic with high accuracy and efficiency.

2. Proposing efficient data structures that operate on switches using only a few Megabytes of memory, significantly reducing the traffic volume between the programmable switch and the sNIC/host.

3. Developing efficient protocols to facilitate communication between the programmable switch and the sNIC/host.

II. High Speed Monitoring Design Challenges

In networks operating at Terabit speeds, it is extremely challenging to monitor traffic at line rate. Even a host running state of the art Network Function Virtualization (NFV) software, can typically only handle 10-100Gbps, meaning ten to one hundred such servers would be needed to accurately monitor a Terabit speed link.

We focus on analyzing the start of each network flow (i.e., examining the first X packets). This has the potential to greatly reduce the volume of traffic that must be inspected, but requires mechanisms to filter traffic from flows that have been deemed benign versus those that require additional analysis. To accommodate such a high volume of traffic at low cost, we propose a hierarchical data plane that makes the best use of a P4 switch, sNIC, and host.

As shown in Figure 1, packets first go through a Bloom Filter on the P4 switch, which provides an efficient way to determine if they come from a flow that has been marked as benign. If so, then no further analysis is needed and the packet can simply be sent out of the switch. If the packet is for a flow not stored in the Bloom Filter, then it must be analyzed by the host to determine if it is suspicious. Instead of immediately sending the packet to the host, its meta-data is stored in a Flow Log stored on the switch. Logs from this are periodically flushed out to the sNIC/ host in a batch to reduce cost. The sNIC maintains a cache of currently active flows, but it does not have sufficient memory to track all flows over a long period of time. When the sNIC faces memory pressure, flows are evicted to the host, which has significantly more memory available to store flow state. Once the sNIC or host have gathered sufficient information about a flow to determine that it is benign, then it must update the Bloom Filter stored on the switch. At this point the flow information no longer needs to be stored by either the sNIC or host, allowing memory in the sNIC and host to be reclaimed.

In the following, we discuss the design challenges for the key components in our system.

A. Aging Bloom Filter

Our design requires a way to white list flows which have been deemed benign. However, there may be millions of such flows in a Terabit scale network. The (relatively) tiny amount of memory available on a switch cannot effectively track such a large list. Bloom Filters offer a good trade-off by significantly reducing the memory usage for set membership tests, with only a small sacrifice in accuracy. However, there are several challenges with implementing a Bloom Filter on a P4 switch. First, a Bloom Filter cannot do removal operations for inactive flows, which means that the filter's bitmap will quickly become full, reducing its accuracy. Yoon et al [11] propose an A^2 buffering Bloom filter to clear the inactive entries while retaining most of the active entries. However, their design does not fit cleanly to the programmable model of a P4 switch, and would require several recirculations which hurts performance. We are investigating the use of an Aging Bloom Filter that adapts their design to the limited memory and pipeline stages available in a P4 switch.

B. Host/SmartNIC Flow Tables

The host and SmartNIC have more memory than the switch, allowing them to maintain state about a larger number of flows. Our design maintains the primary flow table with information about all flows on the host, with a subset of these entries cached on the SmartNIC. When the P4 switch sends a suspicious packet for analysis, the sNIC receives it and checks if it is from a flow stored in its cache. On a cache miss, the Host becomes involved to analyze the packet. As we only care about the start of flows, once the sNIC or Host has been able to successfully analyze X packets for that flow, it will communicate with the P4 switch to update the Bloom Filter with the packet's flow marked as safe. We expect X to be a small number such as 3 if the monitor only seeks to check if a full TCP handshake has been completed, or 7 if it must observe the first few packets exchanged to detect application-level behavior like an HTTP request or SSH login attempt.

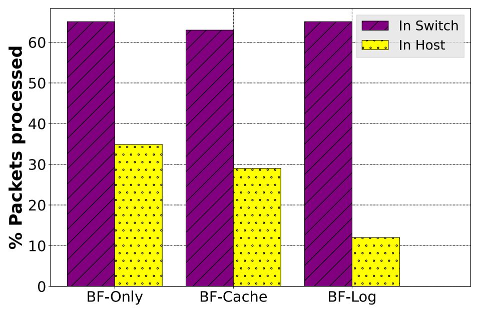

Fig. 2. Packets ratio checked in Switch vs Host

C. Switch/Host Coordination

Our design requires efficient coordination between the switch and sNIC/host. We employ a dynamic bloom filter to identify good flows without sending these flows to the sNIC/host. The bloom filter must be updated by the sNIC/host, making efficient real-time communication crucial. The typical approach for updating switch state is via its control plane interface; however, we find that its update rate is too low. Prior work reports that control plane driven updates can take up to 1 ms [12], yet our design will require us to make hundreds of thousands of such updates every second. To overcome this challenge, we propose a data plane based coordination scheme to increase this speed by several orders of magnitude.

D. Flow Log in Switch

With a straightforward design, every packet unknown to the switch (no information about the flow is available in the switch) must be sent to the sNIC or host. Truncating these packets to only include their header information reduces this cost, but it still can lead to an unacceptably high packet rate for high speed networks. To further reduce the number of packets that must be processed by thesNIC or host, we deploy a Flow Log on the switch. The Flow Log buffers header information from several packets that need to be analyzed, and then transmits them to the sNIC only when the buffer is full. Even a small buffer with space for a few packets can provide a substantial reduction, e.g., a 3 packet buffer reduces the outgoing traffic rate by $2/3$rds but consumes minimal switch resources. Since our analysis is not performed in-line with the traffic, this buffering doesn't incur any extra delay on user traffic.

III. Preliminary Evaluation

In this section, we evaluate the effectiveness of our design through simulation. Our simulator implements the switch and host components of our design, and takes as input PCAP traces which we acquire from the MAWI project [13]. In order to create a higher traffic volume, we merge 750 traces taken from different days, to reach a total packet rate of about 100 Mpps. We require the host to analyze the first 7 packets of flows belonging to common protocols like HTTP(s)/SSH, and the first 3 packets of flows on other protocols. The trace includes a very large number of port scanning slow attack traffic (about 87% of tcp flows have only one packet, potentially indicating a port scan attempt).

Without the switch's help, all packets would need to go to the host. By adding the Bloom Filter to the switch, we can make it so that only the first few packets from each flow need to be analyzed. Figure 2 shows that using only the Bloom Filter reduces the traffic to the host by about 65%. In this case the host only needs to process the headers from the first 3 or 7 packets from each flow, which comes out to 35% of the incoming traffic rate in this trace. We attempt to optimize this further by using the largest possible flowtables on the switch itself (BF-Cache bars in Figure 2). The switch can maintain flow tables that hold up to 262K entries simultaneously. If a packet does not hit in BF, it will either be added if it is not in the flow tables, or it will update the existing entry if it is in the flow tables. If there is a different entry in the flowtables, the old entry will be evicted out to the SmartNIC and the new packet entry will be added to the flowtables. If 3 or 7 packets from a flow hit the cache on the switch, then the switch can directly update the Bloom Filter without ever having to go to the host. This lowers the volume of traffic going to the host to about 29%. However it consumes most of the switch's memory, and thus has a low cache hit rate. Instead, we find that adding a packet log of size 3 for batching (BF-Log bars) provides a better reduction in packets that must be sent to the host, reaching only about 12% of the incoming packet rate. This packet log consumes hundreds of bytes of space compared to several Megabytes for the cache.

IV. Conclusions and Future Work

Our work has demonstrated the potential to allow fine-grained analysis of every flow in the network by leveraging a programmable data plane and focusing on the first few packets of each flow. Our design seeks to overcome the limited memory of a programmable switch by using an efficient Bloom filter and packet log data structure, while offloading the maintenance of flow state to a host or sNIC. In our ongoing work we are seeking to improve the efficiency of our switch-based data structures and the coordination protocols between the host, sNIC, and switch.

This work was supported in part by NICT and NSF Grants 2210380, 2210379.

References

[1] A. Gupta, R. Harrison, M. Canini, N. Feamster, J. Rexford, and W. Willinger, "Sonata: Query-driven streaming network telemetry," in *Proceedings of Anual Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM, Aug. 2018, pp. 357–371. [Online]. Available: http://doi.acm.org/10.1145/3230543.3230555

[2] X. Chen, S. Landau-Feibish, M. Braverman, and J. Rexford, "BeauCoup: Answering many network traffic queries, one memory update at a time," in *Proceedings of Anual Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM, Aug. 2020, pp. 226–239. [Online]. Available: https://doi.org/10.1145/3387514.3405865

[3] J. Sonchack, A. J. Aviv, E. Keller, and J. M. Smith, "Turboflow: information rich flow record generation on commodity switches," in *Proceedings of European Conference on Computer Systems*, ser. EuroSys, Apr. 2018, pp. 1–16. [Online]. Available: https://doi.org/10.1145/3190508.3190558

[4] Z. Zhao, H. Sadok, N. Atre, J. C. Hoe, V. Sekar, and J. Sherry, "Achieving 100Gbps intrusion prevention on a single server," in *Proceedings of USENIX Symposium on Operating Systems Design and Implementation*, ser. OSDI, Nov. 2020, pp. 1083–1100. [Online]. Available: https://www.usenix.org/conference/osdi20/presentation/zhao-zhipeng

[5] Netronome, "The joy of Micro-C," https://cdn.open-nfp.org/media/documents/the-joy-of-micro-c_fcjSfra.pdf, 2014.

[6] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan, "Fast portscan detection using sequential hypothesis testing," in *Proceedings of IEEE Symposium on Security and Privacy*, ser. S&P, May 2004, pp. 211–225.

[7] A. Kuzmanovic and E. W. Knightly, "Low-rate TCP-targeted denial of service attacks: The shrew vs. the mice and elephants," in *Proceedings of Anual Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM, 2003, pp. 75–86. [Online]. Available: https://doi.org/10.1145/863955.863966

[8] H. Namkung, Z. Liu, D. Kim, V. Sekar, and P. Steenkiste, "{SketchLib}: Enabling efficient sketch-based monitoring on programmable switches," in *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 2022, pp. 743–759.

[9] Z. Liu, R. Ben-Basat, G. Einziger, Y. Kassner, V. Braverman, R. Friedman, and V. Sekar, "Nitrosketch: Robust and general sketch-based monitoring in software switches," in *Proceedings of the ACM Special Interest Group on Data Communication*, 2019, pp. 334–350.

[10] S. Panda, Y. Feng, G. Kulkarni, Sameer, K. K. Ramakrishnan, N. Duffield, and L. Bhuyan, "SmartWatch: Accurate traffic analysis and flow-state tracking for intrusion prevention using SmartNICs," in *Proceedings of ACM International Conference on emerging Networking EXperiments and Technologies*, ser. CoNEXT, Dec. 2021. [Online]. Available: https://doi.org/10.1145/3485983.3494861

[11] M. Yoon, "Aging bloom filter with two active buffers for dynamic sets," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 1, pp. 134–138, 2010.

[12] T. Caiazzi, M. Scazzariello, and M. Chiesa, "Millions of Low-latency State Insertions on ASIC Switches," *Proceedings of the ACM on Networking*, vol. 1, no. CoNEXT3, pp. 22:1–22:23, Nov. 2023. [Online]. Available: https://dl.acm.org/doi/10.1145/3629144

[13] "Mawi working group traffic archive." [Online]. Available: http://mawi.nezu.wide.ad.jp/mawi/