Practical Verifiable In-network Filtering for DDoS Defense

Deli Gong*, Muoi Tran*, Shweta Shinde†⋆, Hao Jin‡⋆, Vyas Sekar§, Prateek Saxena*, Min Suk Kang*
*National University of Singapore, {*gongdeli, muoitran, prateeks, kangms*}*@comp.nus.edu.sg*
†University of California, Berkeley, *shwetas@eecs.berkeley.edu*
‡Texas A&M University, *haojin@tamu.edu*
§Carnegie Mellon University, *vsekar@andrew.cmu.edu*

Abstract*—In light of ever-increasing scale and sophistication of modern distributed denial-of-service (DDoS) attacks, recent proposals show that *in-network filtering* of DDoS traffic at a handful of transit networks can handle volumetric attacks effectively. In this paper, we identify a subtle but important security risk in existing in-network filtering proposals. That is, a transit network may use the in-network filtering services as an excuse for any arbitrary packet drops made for its own benefit. For example, a malicious transit network may execute any filtering rules to discriminate against some of its neighboring networks based on its business preference while claiming that it is for the purpose of DDoS defense. We argue that this is due to the *lack of verifiable filtering* — i.e., no single party can check if a transit network executes the filter rules correctly as requested by the DDoS victims. To make in-network filtering a more robust defense primitive, we propose a verifiable in-network filtering system, called VIF, that exploits emerging hardware-based trusted execution environments (TEEs) and offers filtering verifiability to DDoS victims and neighboring networks. Our proof of concept demonstrates that a VIF filter implementation on commodity servers with TEE support can handle traffic at line rate (e.g., 10 Gb/s) and execute up to 3,000 filter rules. We show that VIF can scale to handle larger traffic volume (e.g., 500 Gb/s) and more complex filtering operations (e.g., 150,000 filter rules) by parallelizing the TEE-based filters. As a practical deployment model, we suggest that Internet exchange points (IXPs) are the good candidates to be early adopters of our verifiable filters due to their central locations and flexible software-defined architecture. Our large-scale simulations of two realistic attacks (i.e., DNS amplification, Mirai-based flooding) show that adopting VIF filtering service at only a small number (e.g., 5–25) of large IXPs is sufficient to handle the majority (e.g., up to 80–90%) of DDoS traffic.

I. Introduction

Distributed denial-of-service (DDoS) attacks are highly prevalent, globally accounting to more than 20,000 attacks per day [21]. In the last decade, new attack strategies such as amplification [64] and new attack sources such as IoT devices [80] have surfaced, which have resulted in attacks of extremely high volume [78].

A large number of DDoS defenses have been extensively studied over the past two decades. Among them, an effective defense against the ever-increasing scale of DDoS attacks is *in-network filtering* or empowering DDoS victim networks to install in-network traffic filters in the upstream transit networks. This idea was proposed in early efforts (e.g., Pushback [50], D-WARD [53], AITF [4]) and has repeatedly resurfaced in standardization committees (e.g., see the recent DDoS Open Threat Signaling [55]).

⋆ Research done while working at National University of Singapore.

Dropping suspicious packets closer to the attack sources at the requests of DDoS victims is desirable because it (1) reduces wasted traffic on downstream ISPs, thereby reducing overall network usage incurred by routing malicious traffic; and (2) has the potential to handle increasing attack volumes (e.g., several Tb/s) as the volume at each distributed filtering point is much lower than the aggregate volume at the victim [4].

Unsurprisingly, there is renewed interest in the research community on revisiting in-network filtering solutions. Indeed, a recent DDoS defense architecture, called SENSS [63], suggests that the traffic filters installed at a few large transit ISPs directly by the remote DDoS victims can prevent most of the volumetric attack traffic from flooding the victim networks. Also, an economic compensation model proposed in SENSS allows the DDoS victims to pay the transit networks for the requested filtering tasks.

In this paper, we identify that the adoption of in-line filtering introduces a subtle but important shift in how we perceive and handle packet drops in the Internet, which, unfortunately, leads to a class of new network attacks. In a legacy transit network (i.e., no in-network filtering), packet-drop events are considered as faults, such as network failures or congestion; see the secure network fault localization proposals [8], [85]. In contrast, when packets are dropped in a transit network with an in-network filtering service, it is unclear to neighboring networks whether the drops are the results of network faults or a legitimate in-network filtering operation on behalf of a remote DDoS victim. With this ambiguity, a malicious transit network can now execute arbitrary traffic filtering for its own benefit with impunity and use DDoS defense as an excuse for its packet filtering. Since packet drops may occur without any network faults, network fault localization also becomes less useful. As illustrated in Figure 1, when a transit network is requested to execute a filter rule set by a DDoS victim, it can arbitrarily modify the filter rules to discriminate against some of its neighboring networks; e.g., dropping the majority of traffic from neighboring network A but no filtering for neighboring network B. Detecting such filter rule violations is not straightforward because neighboring networks or a victim network cannot individually reason about the correct filtering execution. This is a practical concern considering the already-existing disputes between transit networks (e.g., a dispute between Level3 and Comcast [83]) in today's competitive transit market [27].

We argue that the *verifiability* of in-network filtering mitigates such misbehaviors by malicious filtering service providers. With filtering verifiability, when a filtering network executes a manipulated filter rule, the DDoS victim network who requested the filter rule or the direct neighboring au-

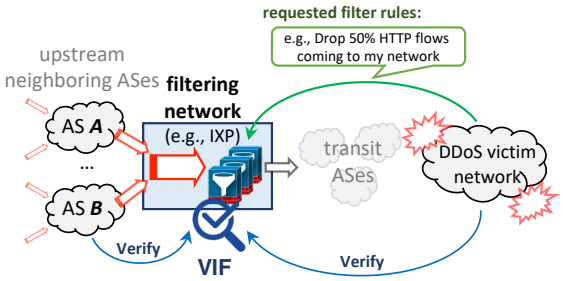


Fig. 1: Example of in-network filtering in a transit network. VIF enables the direct neighboring networks or the DDoS victim to *verify* if the filtering network executes the requested filter rules correctly.

tonomous system (ASes) of the filtering network can detect the misbehavior individually.

In this paper, we offer a technical mean for enabling verifiability of filtering operations and develop a practical and scalable in-network filtering system, called VIF.[1] VIF has a generic architecture designed for any transit networks (e.g., Tier-1, large Tier-2 ISPs, IXPs). We exploit software networking functions running on commodity hardware with trusted execution environments (TEEs), such as Intel SGX [17] and Arm TrustZone [5], mainly for the integrity of the network function executions. TEEs are widely available in today's commercial off-the-shelf (COTS) server platforms; see Microsoft and Google's SGX-based cloud platforms [60], [65].

We identify two key technical challenges to realizing the VIF vision in practice:

- **Auditability.** Making the overall traffic filtering operations auditable is the key to the verifiable in-network filtering. However, despite the TEE-based strong integrity guarantees for filtering functions, achieving auditable filtering operations is not trivial. One challenge is that the correct filtering operations are heavily affected by *external* inputs to the filter (e.g., incoming packet order, time clock feeds), which can be controlled by a malicious transit network. Another challenge is that a malicious transit network can *bypass* our VIF filters by reconfiguring its network and avoid using the filters for adversary-selected packets.
- **Scalability.** With the growing size of DDoS attacks, VIF should be able to scale out its filter capacity in terms of the bandwidth and the number of filter rules. However, the parallelization with multiple TEE-based auditable filters is not straightforward because some necessary network components for parallelization (e.g., traffic load balancers) are not directly auditable. Moreover, distributing filter rules across multiple auditable filters creates an optimization challenge, which involves two-dimensional resource constraints.

Approach and Contributions. VIF design makes the following key contributions to address these aforementioned challenges:

- We analyze the requirements for auditable filters, particularly, their reliance on the external inputs to the filters, which can be controlled by malicious filtering networks (§III-A). Our key insight is that it is sufficient that traffic filters are *stateless* for auditable filter operations. We also implement an effective *bypass detection* that relies on accountable packet logs (with an efficient sketch implementation) measured inside the TEE (§III-B). The packet logs can be used to identify packet drops/injections made outside of the auditable filters. We demonstrate that an efficient *line-rate* implementation (nearly 10 Gb/s throughput performance) of the auditable traffic filters with the TEE support is possible with several system optimizations (§V). Our proof of concept of the VIF filter is open source and available online[2]; and
- For highly scalable filtering architecture, we implement a dynamic filter rule distribution algorithm across multiple auditable filters and untrusted network components [28], [58] (§IV). We implement a heuristic that can quickly reconfigure a large number of filter rules (e.g., 150,000 filter rules) and a large volume of incoming traffic (e.g., the total volume of 500 Gb/s) with auditable filter instances (e.g., 50 filters).

As a practical deployment path, we argue that major Internet exchange points (IXPs) are the promising adopters of VIF (§VI). In the last decade, IXPs have become the central infrastructure of the global Internet connectivity [1], [12], with large IXPs handling daily traffic volumes comparable to those carried by the largest Tier-1 ISPs, which makes the IXPs the perfect candidates for our verifiable filtering service [22]; see the recent prototype deployment of a DDoS filtering service in one European IXP [23]. We perform large-scale simulations with two realistic attacks (i.e., DNS amplification, Mirai-based DDoS attacks) and show that deploying VIF in a small number (e.g., 5–25) of large IXPs is enough to handle the majority (e.g., up to 90%) of DDoS attacks (§VI-C).

II. Problem Definition

In this section, we describe the threat model we consider in this paper (§II-A), the desired properties of the VIF design (§II-B), the trusted execution environment model (§II-C) and the assumptions we make in this work (§II-D).

A. Threat Model

Our threat model focuses on the problem of a single potentially malicious transit network that offers in-network filtering services but manipulates the filter rules submitted by the DDoS victim, which we call as *filter rule violation attacks*. In general, let us consider a filter rule R requested by a victim network. The malicious filtering network may change R arbitrarily into a different filter rule R' and apply it to all traffic destined to the victim network. Also, the malicious filtering network may apply different modified rules $R'_1, R'_2, \cdots$ for different traffic flows (e.g., packets delivered via different neighboring ASes).

We present two example attack goals, where the manipulation of filter rules at a filtering network can seriously disrupt the packet forwarding services for the neighboring ASes and the remote DDoS victim networks. In both attacks, we use the same example illustrated in Figure 1, i.e., the

[1] VIF stands for 'Verifiable In-network Filtering'.

[2] https://github.com/InNetworkFiltering/SGX-DPDK.

filtering network manipulates the DDoS-victim-submitted filter rule R =[Drop 50% of HTTP flows destined to victim network].

[Goal 1] Discriminating against some neighboring ASes. ASes expect their traffic to be reliably forwarded by the transit networks. Yet, when a transit network offers in-network filtering services, it can silently differentiate the quality of packet forwarding for different neighboring ASes. In particular, the filtering network can apply the modified filter rules for each neighboring AS based on its own business preference. Instead of applying the same original rule R, the filtering network may apply R'_A =[Drop 20% of HTTP flows destined to victim network] for traffic flows delivered by AS A and R'_B =[Drop 80% of HTTP flows destined to victim network] for traffic flows delivered by AS B. Such discriminatory filtering is hard to detect because individual neighboring AS A and B do not know the unmodified rule R and, even if they know R, they cannot determine if the traffic filter applied to their packets is R. Each neighboring AS may try to infer the packet drops of the end-to-end path indirectly (e.g., via monitoring TCP sessions); yet, it is *insufficient* because pinpointing the exact location of packet losses (a.k.a. fault localization) is known to be hard without large-scale network collaboration [3], [8].

[Goal 2] Reducing operational cost with inaccurate filtering. To reduce the operational cost, the malicious filtering network may violate the filter rules submitted by the DDoS victims. For instance, consider that the malicious filtering network wants to use *only* 10 Gb/s of its filtering capacity for the rule R while total incoming traffic of 50 Gb/s should be sent to the filters. To achieve this, the filtering network can send only 10 Gb/s traffic to its filters and execute the unmodified rule R. For the rest of the 40 Gb/s traffic, however, the filtering network can simply allow or drop all *without* using the filtering capacity. Since the victim network has no information about the incoming traffic arrived at the filtering network, it cannot directly detect this attack.

B. Desired Properties: Filtering Auditability and Scalability

We have the two desired properties of the VIF design. First, VIF aims to have the *filtering auditability* and remove the attack capabilities required for the filter-rule violation attacks. A filter rule R is said to be auditable if any modification of the rule R and its execution by the filtering network can be detected by the victim networks or the direct neighboring ASes. Second, our VIF system must be easily *scaled up* because DDoS attacks are increasingly scalable in the number of attack flows and the total bandwidth. We have observed an escalation in the volume of DDoS attack traffic [74], [80] and attacks are getting more sophisticated; e.g., multi-million bots are becoming more common [52].

C. Trusted Execution Environment with Intel SGX

VIF uses TEEs, particularly Intel SGX in this work, as the feasible hardware-based root of trust. Intel Software Guard Extensions (SGX) is a recent architectural feature that allows secure execution of a program on a computing infrastructure in control of an adversarial operator [17], [51]. SGX also supports secure execution of a user-level program with no modification of underlying commodity software stacks [6], [9], [70]. In particular, it offers the isolated execution of the application logic in a protected memory region, called an *enclave*, which prevents the operator from tampering it. Moreover, it supports remote attestation that allows a third party to audit that the correct application and data has been loaded in an enclave. The attestation process starts when a verifier issues an attestation challenge to the enclaved application. The enclave then provides a report, which is cryptographically signed with the attestation key of the SGX hardware. Next, the attestation report is verified by the Intel Attestation Service (IAS), which is distributed globally [41]. Alternatively, Intel also allows *anyone*, who gets a certificate from Intel, to run their own remote attestation services and verify the attestation report [66].

D. Assumptions

We assume an out-of-band channel between the victim network and the filtering network that is available even when the victim network is under DDoS attacks.[3] We also consider a typical DDoS attack scenario where the victim network is congested but its upstream ISP networks are still available [4].

We assume that ISPs (e.g., victim networks) trust the remote attestation process for the integrity guarantees of the VIF enclave. We also assume an idealized implementation of VIF that has no backdoor. We leave a formal verification of VIF implementation as future work. Hardware and side-channel attacks (e.g., [33], [38], [47]) are out of the scope of this paper since countermeasures to these (e.g., [14], [18], [34], [68], [69], [73]) are orthogonal to the design of VIF.

III. Auditable Filter Design

The VIF filtering operation is enclosed by an SGX enclave where the integrity of its execution is guaranteed, i.e., a malicious filtering network cannot tamper it. Furthermore, the filtering internal logic and states are also securely verified via the remote attestation process [41]. The isolated execution and remote attestation are useful in realizing the auditable filter; yet, they are insufficient because (1) the filtering decisions can be influenced by the external inputs to the filter such as packet order and time clock feeds, which are controlled by the filtering network; and (2) the malicious filtering network may redirect the traffic within its network to bypass the filtering operations.

To address these two challenges, we suggest that the filtering operations be *stateless* and hence be independent from the external inputs (§III-A) and implement the enclaved packet logs to detect bypassing attempts (§III-B).

A. Stateless Filter Design

We first describe an abstract model for our enclaved filter f to analyze the dependencies of the enclaved filters:

$$\{\texttt{ALLOW}, \texttt{DROP}\} \leftarrow f\big(\langle p, a\rangle, (\langle p_1, a_1\rangle, \langle p_2, a_2\rangle, \cdots)\big), \quad (1)$$

where $\langle p_i, a_i\rangle$ denotes that packet p_i arrives at the enclaved filter at time a_i (measured by the enclave's internal clock),

[3] ISPs traditionally have maintained out-of-band channels (e.g., external email servers, telephone lines [55]) for inter-ISP communication.

$\langle p, a\rangle$ represents the packet p that is being evaluated and its arrival time a, and the following time relationship holds $a > a_1 \geq a_2 \geq \cdots$.

Notice that in this abstract model, the filtering operation of a packet p depends on the packet arrival time and the order of the packets, which can be exploited by the filtering network. Here, we summarize the two properties that are needed to make VIF filter auditable:

- *Arrival-time independence.* The filtering decision should be independent of packet-arrival time because it can be easily manipulated by a malicious filtering network (e.g., delaying individual data packets). Moreover, a malicious filtering network can delay the time query/response messages to/from the trusted clock source for the enclave [40], slowing down the enclave's internal time clock.
- *Packet-injection independence.* The filtering decision should not depend on the previous packets since a malicious filtering network can also inject any arbitrary packets into the traffic flow and influence the filtering decision.

Thus, to ensure that the filtering operations are auditable, the filtering function f can be *independent* of all the previous packets and their arrival times; that is,

$$f\big(\langle p, a\rangle, (\langle p_1, a_1\rangle, \langle p_2, a_2\rangle, \cdots)\big) = f(p), \tag{2}$$

which simplifies the filter design to n-tuple (e.g., `srcIP, dstIP, srcPort, dstPort, protocol`) per-packet filters. In other words, the filtering decision of packet p solely relies on p, e.g., five-tuple bits. Such a simple *stateless* n-tuple filter design has its own limitations (e.g., incapable of handling complicated application-level DoS attacks); yet, the stateless filters are sufficient for handling the majority of large volumetric attacks, e.g., more than 75% of DDoS attacks [21].

Particularly, for handling volumetric attacks, we allow victim networks to express filter rules for exact-match five-tuple flows (e.g., a specific TCP flow between two hosts) or coarse-grained flow specifications (e.g., HTTP connections from hosts in a /24 prefix); see Appendix A for several practical design points for our enclaved filter.

B. Filter Bypass Detection

The auditability of the VIF filter guarantees that the filter operates correctly for the given packets from a filtering network to a victim network. However, packets may not be properly filtered when a malicious filtering network configures the traffic to *bypass* the VIF filter, hence violating the filter rules. Particularly, the manipulation of the traffic flows happen *outside* of the protected enclave and thus cannot be detected by the auditable VIF filter itself. We categorize the filter bypass attacks as follows:

- *Injection after filtering:* The VIF filter drops a packet p but the adversary injects a copy of p into the packet stream that is forwarded to the victim;
- *Drop after filtering:* A packet p is allowed by the filter but the adversary drops p before forwarding it to the victim; and
- *Drop before filtering:* The filtering network drops a packet p even before it is processed by the filter.

Note that, we do *not* consider *injection before filtering* operations by a filtering network as an attack because it does not affect the filtering decision due to the *packet-injection independence* property of the filters (see Section III-A).[4]

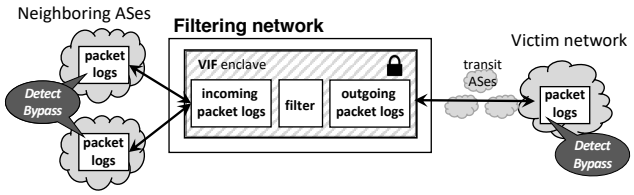


Fig. 2: Neighboring ASes and victim network individually detect filter bypass attacks. VIF uses an efficient sketch data structure for packet logs.

Bypass Detection. We allow the victim network and the neighboring ASes of the filtering network to detect such bypass attempts individually by implementing the *accountable packet logs* inside the enclave for incoming and outgoing packet streams, see Figure 2. For each packet log, we utilize a sketch, particularly a *count-min sketch*, a memory-efficient data structure that stores summaries of streaming data [16]. With the sketch-based packet logging, the VIF filter keeps only the measurement summary inside an enclave and significantly minimizes the memory footprint; e.g., less than 1 MB per sketch. With some additional data-plane optimizations (see Section V-A), the computational overhead of computing two sketches per packet is negligible (see Section V-B).

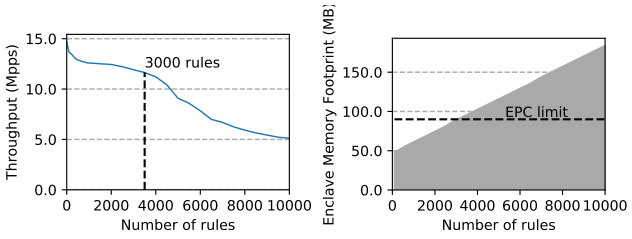
To detect bypass attempts by the filtering network, the victim network queries the authenticated outgoing packet logs from the VIF enclave and compares it with its own local sketch. Since the count-min sketch logs do not record non-existent packets (i.e., no false negative), any discrepancy between the two sketches implies *injection after filter* and/or *drop after filter* attacks by the filtering network. The computation and bandwidth overhead for the logs queries is negligible; i.e., sketching is highly efficient and requires sending only a few MBs of sketch memory via the already established channel with the victim network.[5] Similarly, individual neighboring ASes of the filtering network can detect the *drop before filtering* attacks by comparing their own local packet logs with the authenticated incoming packet logs of the VIF filter.

Handling misbehaviors. When the victim network detects any bypass attempt, it can decide to abort the ongoing filtering request with the filtering network. In practice, the VIF filtering network should allow a short (e.g., a few minutes) time duration for each filtering round so that victim network can abort any further request quickly when it detects any bypass attempts. The neighboring ASes can choose another downstream network when they obtain the evidence that their current downstream network offers filtering services but intentionally drops their packets before they reach the VIF filters.

Handling malicious intermediate ASes. The bypass detection mechanism may cause *false positives* when some packets are dropped after leaving the VIF filter but before reaching the

[4]Moreover, the detection of packet injections before the enclave operations is hard without explicit coordination with traffic sources.

[5]The computational overhead of the victim network should also be low since it only requires an efficient sketch on a commodity server without SGX overhead.



(a) Filter throughput (b) Filter memory footprint

Fig. 3: Filter throughput degradation with the increasing number of filtering rules.

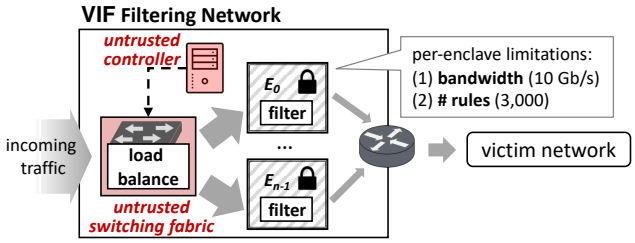


Fig. 4: Scalable VIF architecture. Multiple VIF enclaves are parallelized with an untrusted load balancer.

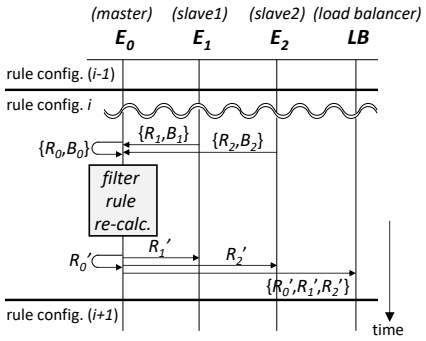


Fig. 5: Protocol for filter rule recalculation and redistribution across three enclaved filters: E_0, E_1, and E_2.

victim network. When this happens, the victim network cannot accurately pinpoint where the packet drop has happened [3], [8]. The packet could have been dropped by one of the *intermediate* transit networks between the VIF filtering network and the victim network, or by the VIF filtering network itself.

Therefore, instead of locating such packet drops, VIF allows the victim network to dynamically *test* all the intermediate ASes (an inter-domain path usually have only 3–6 ASes) by rerouting its inbound traffic to avoid each of ASes being tested in a short time using the well-known BGP poisoning-based techniques (e.g., [42], [72]). BGP poisoning does not require network collaboration and can detour the traffic in only a few minutes [71], [72], [77]. We describe the detailed test steps in Appendix B.

IV. Scalable Filter Design

In this section, we first analyze the performance (e.g., throughput, network I/O) bottlenecks of a single auditable filter (§IV-A) and then describe a scalable filtering design with multiple enclaved filters running in parallel and an untrusted load balancer (§IV-B).

A. Bottlenecks: Maximum Bandwidth and Number of Rules per SGX Filter

Recent works such as mbTLS have demonstrated that the 10Gb/s performance per enclave can be reached with a four SGX cores machine [56]. Although the processors with six or more cores available on the market[6] may support larger bandwidth, we consider the maximum network I/O performance of each SGX enclave is 10 Gb/s in the rest of the paper.

Since the SGX-based filter must match the installed rules with incoming flows to perform filtering, the number of filter rules naturally becomes the bottleneck of the filter's performance. Indeed, we measure the throughput of traffic processed by a single enclaved filter with different numbers of filter rules and show the results in Figure 3a. We can see from Figure 3a that when the number of filter rules exceeds approximately 3,000, the VIF filter's throughput performance rapidly degrades.

One of the explanations is that, when the number of filter rules increases, the lookup table for the packet processing inside an SGX enclave also grows accordingly. Even when we use the state-of-the-art multi-bit tries data structure for looking up the filter rules (see Section V for details), the memory size of the lookup table still grows linearly with the number of filter rules, as shown in Figure 3b. This result also confirms that the Enclave Page Cache (EPC) limit is around 92 MB, as seen in many other works (e.g., [45]).

[6]List of SGX-enabled processors is available at: https://ark.intel.com.

B. Scalable Filtering with Multiple SGX Filters

Given the architectural limitation of secure computing resources in currently available SGX architecture, the single-enclave filtering deployment may not be able to deal with the increasing attack volume and number of attack flows. Hence, we propose a generic VIF architecture that can easily scale up as the number of filters grows, as shown in Figure 4. The scalable VIF design includes multiple enclaved filters running in parallel and some *untrusted* facilitating components such as the high-bandwidth switching fabric and the controller.

Our multiple SGX filters design is robust against attacks by the untrusted load balancer that may redirect to a filter the traffic flows that do not match with the filter rules assigned for that filter. Our individual trusted filter can easily detect such a misbehavior by comparing the packets it receives with the assigned rules. Also, if the load balancer drops the traffic flows that are supposed to be redirected to an enclave, it can be detected by the bypass detection of the auditable filters (see Section III).

Next, we describe how filter rules are distributed and dynamically adjusted among multiple enclaves.

Filter rules distribution protocol. Since the traffic flows being filtered are frequently changed, the filter rules also need to be updated and redistributed among the filters accordingly.

We consider that the distribution of the filter rules happens in *rounds*, i.e., the entire filter rule set is known and does not change until the next rule reconfiguration is executed. In each round, the filter rules are calculated and redistributed via a simple master-slave topology among multiple enclaved filters. We illustrate the protocol in Figure 5, where we have filter E_0 as the master node and E_1, E_2 as the slave nodes. In particular, when a reconfiguration of filter rules is desired (e.g., traffic volume or the number of filter rules handled by a certain filter exceeds a threshold), any enclaved filter may initiate a rule redistribution round and become the master node. Then, all the slave nodes upload their filter rule sets (R_i for E_i) and the array of the average received flow rates of each rule set R_i (B_i for E_i) to the master node. The master node calculates re-configured filter rules, which then are redistributed to all the slave nodes and the load balancer. If the calculation requires changes to the number of enclaves, necessary additional steps (e.g., creating and attesting more enclaved filters) may be required before the rule redistribution.

Filter rules calculation optimization problem. In each filter rules redistribution, the master node has to allocate the bandwidth and rules to all enclaved filters. We consider the calculation of the optimal rule sets for each filter enclave as solving a mixed integer linear programming (ILP) optimization problem. We assume k filter rules as r_i $(1 \leq i \leq k)$ and the corresponding incoming bandwidth as b_i $(1 \leq i \leq k)$.[7] A single enclave has a memory limit M (e.g., 92 MB) and a bandwidth capacity G (e.g., 10 Gb/s) as we have discussed in Section IV-A. Then, we can decide the minimum number of enclaves as needed as $n_{min} = \lceil \max\left(\frac{1}{G}\sum_{i=1}^{k} b_i, \frac{ku}{M-v}\right)\rceil$. To allow some room for optimization, the number of enclaves is taken as $n = \lceil \max\left(\frac{1}{G}\sum_{i=1}^{k} b_i, \frac{ku}{M-v}\right) \times (1+\lambda)\rceil$ where $\lambda \geq 0$ is an adjustable parameter for additional enclaves. We define real-valued variables $x_{i,j\ (1\leq i\leq k, 1\leq j\leq n)}$ denoting the portion of bandwidth b_i allocated to the j-th enclave, and binary variables $y_{i,j\ (1\leq i\leq k, 1\leq j\leq n)}$ representing if rule r_i is installed on the j-th enclave (i.e., $y_{i,j} = 1$). Based on the allocation plan indicated by $x_{i,j}$ and $y_{i,j}$, we consider $C_j = u \times \sum_i y_{i,j} + v$ as the memory cost function, which is a linear function of the number of rules installed (where u, v are constants). Also, $I_j = \sum_i x_{i,j} y_{i,j}$ is considered as the allocated bandwidth. We present the detailed ILP formulation in Appendix C.

Greedy algorithm to calculate filter rules. Solving the above-mentioned optimization problem is inherently costly when $k \times n$ is large (e.g., $> 10K$). Thus, we propose a greedy algorithm (see Appendix D) that finds a sub-optional solution within a reasonably short time period. The high-level intuition of the greedy algorithm is to pre-compute the two parameters—(1) the number of rules per enclave h and (2) the bandwidth quota per enclave g—and arrange the rules and bandwidths for the obtained two parameters heuristically.

[7] We denote b_i as the incoming bandwidth measured for a filter rule for easier understanding. In practice, each enclave would produce byte counts without timestamping them because their individual clock sources are untrusted (see Section III-A). The byte counts are then collected in a timely manner and used for the optimization problem.

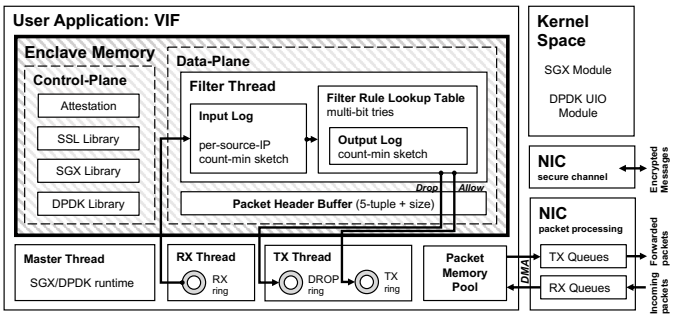


Fig. 6: VIF architecture.

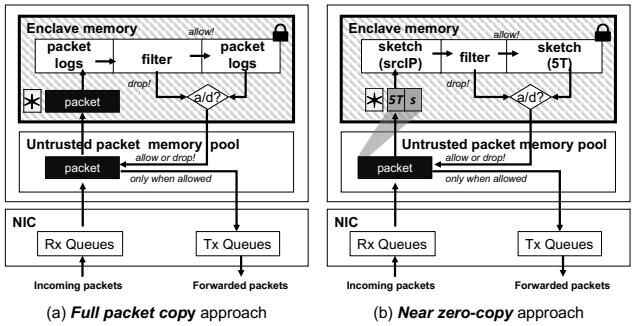


Fig. 7: Two packet copy approaches for the auditable filter and packet logs. In the near zero-copy approach, we copy only the memory reference ($*$), the five-tuple ($5T$), and the size (s) of the packet into the enclave.

V. Implementation and Evaluation

We implement a proof of concept of the VIF filter using SGX and various optimizations (§V-A) and then evaluate its data-plane (§V-B) and scalability performance (§V-C) for large attack volume and number of attack flows.

A. Implementation

Overview. We build the VIF filter as a Linux userspace application with Intel SGX SDK 2.1 and DPDK 17.05.2 for high-speed packet processing. Figure 6 shows the main components of the VIF filter and the minimal trusted computing base (TCB) of code and data inside the enclave, which includes the entire control-plane and the key parts of the data-plane logic (e.g., packet logging and filtering). The control plane performs remote attestation and manages the keys for communication with a DDoS victim. The design of the data plane follows DPDK pipeline model, where three threads (i.e., RX thread, Filter thread, and TX thread) run on individual CPU cores and packets are passed between cores via DPDK lockless rings (i.e., RX ring, DROP ring, and TX ring). Every thread runs a small loop polling the hardware or software buffers in the previous stage, processes a batch of the packets, and passes it to the next stage in the pipeline.

Optimization: Near zero-copy design. For every incoming data-plane packet, a VIF filter logs the packet, filters it based on the given filter rule set, and logs it again if it is allowed by the filter, as shown in Figure 7. Figure 7(a) shows a naive approach, where a VIF filter makes the entire copy

of incoming packets into the enclave and operates these functions over the packet copies inside the enclave. This *full-packet copy* approach can be considered as the baseline packet processing mechanism of other existing SGX-based middlebox applications (e.g., Tor nodes [43], [70], TLS middleboxes [32], [39], [56], inter-domain routing [15], [44], and IDSs [32], [67], [76]), where secure operations over the full packet bytes are required (e.g., full packet read or encryption). However, this approach may incur too much overhead when performing line-rate processing due to the remaining EPC memory for a VIF filter is only about 92 MB.

We thus minimize the dynamic memory usage and avoid the paging by copying only certain header fields into the enclave, which we call *near zero-copy* optimization. This allows more memory space for filter rules and the lookup table. In particular, only a fraction of each packet's header fields (i.e., the five-tuple fields, $5T$, and the packet size, s) are copied into the enclave memory along with the memory reference ($*$) of the packet, as shown in Figure 7(b).[8] The copied data $\langle 5T, s \rangle$ represents the packet and is used for the logging functions and the filter operation. The memory reference $*$ is used to perform the corresponding operation (e.g., allow or drop) for the packet in the untrusted memory pool.

With the copied five-tuple and the size, we first log each packet using a count-min sketch [16] (with two independent linear hash functions, $64K$ sketch bins, and 64-bit counters) for memory efficient (e.g., 1 MB) per-source-IP counters. The per-source-IP sketch for the incoming packets enables each neighboring ISPs of the filtering network to detect the 'drop before filtering' bypass attack discussed in Section III-B. For forwarded packets, we also record another count-min sketch based on the full 5-tuple bits so that the victim network can detect bypass attempts. The latencies increased by the two sketch operations are negligible because only four linear hash function operations are conducted in the data-plane path. Each counter has 64 bits and takes only around 1 MB EPC memory per instance of the count-min sketch.

Optimization: Reducing the number of context switches. Another major overhead stems from the context switches when user application calls the enclave functions (ECall) or the enclaved function calls the outside functions (OCall). We address this performance degradation by reducing both types of calls in the filter thread: (1) VIF only needs one ECall to launch the filter thread and initiates its polling; and (2) the filter thread makes no OCalls as the communication with other threads relies only on the software rings.

Trusted computing base (TCB). Beyond the DPDK library containing about 64K lines of code (LOC), our VIF filter contributes to the TCB only 1,206 LOC which includes the modification of DPDK `ip_pipeline` (1044 LOC) and the packet logging and near zero-copy functions (162 LOC).

B. Line-rate Data-plane Performance

Testbed Setup. We test our implementation with two machines: one is a packet generator and one deploys VIF filter.

[8]Such reduction of byte copies is allowed for our auditable filter applications but this does not necessarily apply to any other SGX-based middlebox applications.

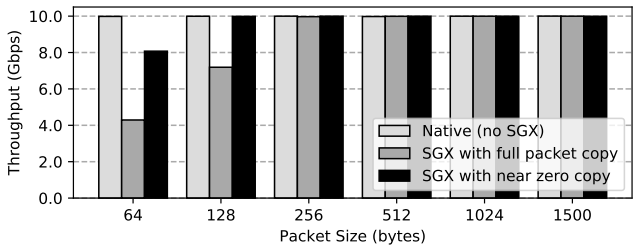


Fig. 8: Throughput performance in bit-per-second for varying packet sizes and 3,000 rules with three implementation versions: (1) Native (no SGX), (2) SGX with full packet copy, and (3) SGX with near zero copy.

The packet generator has an Intel E5-2630 v3 CPU (2.40 GHz, 8 cores) and 32 GB memory. The filtering machine has an Intel i7-6700 CPU (3.40 GHz, 4 cores) and 8 GB memory. Both have 10 GbE Intel X540-AT2 network cards and run Ubuntu 16.04.3 LTS with Linux kernel 4.10. On the packet generator machine, we use `pktgen-dpdk` 3.4.2 [24] to generate the traffic saturating the 10 Gb/s link between the two machines.

Throughput performance. We benchmark the maximum throughput performance of the filter with the packet size of 64, 128, 256, 512, 1024, and 1500 bytes for three different versions of the VIF filter implementations: (1) native filter without SGX, (2) SGX-based filter with full packet copy, and (3) SGX-based filter with near zero-copy.

Figure 8 shows the throughput performance for varying packet sizes for the three implementations. For the packet sizes of 256 Byte or larger, all the three implementations achieve the full line-rate of 10 Gb/s. With small packet sizes, however, we observe some degradation due to the use of SGX. Particularly, when we make full packet copies for each incoming packet, the filter experiences significant throughput degradation. The near zero-copy implementation demonstrates 8 Gb/s throughput performance even with 64 Byte packets and 3,000 filter rules. Additionally, we present the experiment results of VIF evaluation in packet per second metric in Appendix E.

Latency performance. We also measure the latency for the near zero-copy version with various packet size starting from 128 bytes. The results are 34 μs (128 bytes), 38 μs (256 bytes), 52 μs (512 bytes), 80 μs (1024 bytes), 107 μs (1500 bytes). All the measurements are average latency over 10-second run with 8 Gb/s constant traffic load, which are reported by `pktgen`'s latency measurement function.

Connection-preserving filtering performance. We evaluate the detailed performance of connection-preserving filtering. We present the result in Appendix F.

Remote attestation performance. Our detailed remote attestation performance can be found in Appendix G.

C. Scalable Filter Rule Distribution

We evaluate the solving performance of the mixed ILP optimization problem described in Section IV-B with the CPLEX solver [20] in a server-grade machine with 20 cores. We use 3,000 or more filter rules that would cause the throughput degradation of each VIF filter. In this evaluation, we consider

TABLE I: Execution times for the ILP solution and the greedy algorithm solution. The CPLEX's mixed ILP solver is configured to stop when sub-optimal solutions are found.

Number of rules (k)	CPLEX (sub-optimal)	Greedy
5,000	210.49s	0.31s
10,000	772.43s	0.50s
15,000	1,614.96s	0.73s

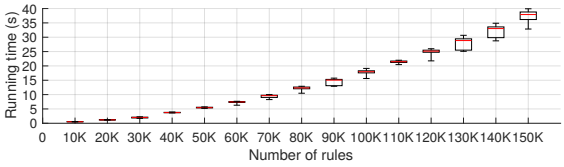


Fig. 9: The time taken to complete the heuristic algorithm for optimizing filter rules across multiple enclaves with varying number of rules k. Red bands indicate the median.

that the total traffic rate going through the entire VIF filter is 100 Gb/s. The incoming traffic distribution across the filter rules follows a lognormal distribution.

With the number of rules more than 3,000 and the number of enclaves more than 10, we find that the CPLEX's mixed ILP solver cannot return the optimal solutions within any reasonable time period. To evaluate the effectiveness of our greedy algorithm in Section IV-B, that is, how close it is to the optimal solutions, we use a small number of filter rules ($10 \leq k \leq 15$) and confirm that the difference between the optimal cost function calculated by the CPLEX's mixed ILP solver and the results from our greedy algorithm is only 5.2%.

We now compare the execution time of the CPLEX's mixed ILP solver and our greedy algorithm when the number of rules is between 5,000 and 15,000 and show the results in Table I. To measure the execution time of the CPLEX's mixed ILP solver for our optimization problem, we configure the solver to stop earlier when it finds a first, sub-optimal solution. As shown in Table I, the CPLEX solver even requires about 200 – 1,600 seconds to find the sub-optimal solutions, which are unacceptably slow for the dynamic filter rules redistribution operations. On the other hand, our greedy algorithm runs three orders of magnitude faster than the CPLEX solver with the same number of filter rules.

Figure 9 shows the extended experiments on the execution times of the greedy algorithm for varying number of rules at a much larger range. We also run this experiment with the total traffic bandwidth of 500 Gb/s, which follows the log-normal distribution. In all the range we test (10K–150K filter rules), the greedy algorithm requires no more than 40 seconds. This enables a near real-time dynamic filter rule re-distribution for large numbers of VIF filters.

VI. Practical Deployment at IXP

VIF has a generic architecture designed for any transit networks (e.g., Tier-1 or large Tier-2 ISPs); yet, as the first deployment model, we suggest to deploy it in major Internet exchange points (IXPs). In this section, we present why IXPs, among other transit infrastructure, are the good candidates to deploy our verifiable in-network filtering (§VI-A). We also provide a deployment example of VIF at an IXP (§VI-B) and then evaluate the effectiveness of VIF at IXPs against DDoS attacks with two real attack source data (§VI-C). Finally, we provide a simple cost analysis for deploying VIF service at an IXP for filtering up to 500 Gb/s traffic (§VI-D).

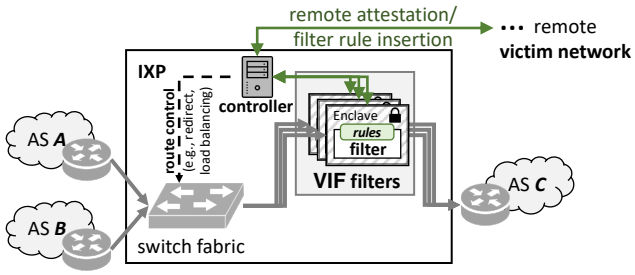


Fig. 10: Deployment example of VIF at IXP.

A. Internet Exchange Points (IXPs)

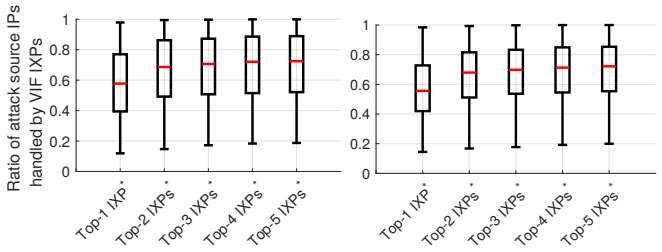
IXPs are physical locations where multiple autonomous systems (ASes) peer and exchange traffic and BGP routes. Essentially, an IXP is a large layer-2 fabric and connects ASes (e.g., ISPs, content providers, cloud providers) in close proximity. IXPs provide great convenience to ASes in making peering relationship with many (e.g., hundreds or thousands) other ISPs without the hassle of individual bilateral agreements. The Internet currently has more than 600 globally distributed IXPs [11] and some large IXPs serve multi-Tera b/s traffic volume, which is comparable to large Tier-1 ISPs [1], [12].

Recently, as more video content providers and large cloud providers rely on IXPs for a lower cost but faster transit of their traffic, emerging *value-added services* are expected from IXPs; e.g., application-specific peering, inbound traffic engineering, wide-area server load balancing, and redirection through middleboxes [37]. New innovation for these value-added services has been possible because IXPs have a flexible architecture model (especially, compared to traditional transit networks, such as ISPs). An IXP usually consists of a single data center in a single physical facility; thus, software-based architecture available for data centers can be easily adopted; see software-defined IXPs [35]–[37].

Due to their topological centrality, however, IXPs unfortunately often suffer from collateral damage when other networks are targeted by DDoS attacks [61]. Worse yet, IXPs are sometimes directly targeted by DDoS attacks; see an attack incident in 2016 against multiple IXPs: AMS, LINX, and DE-CIX [10]. A traffic filtering service could easily be a natural next value-added service for IXPs [22].

B. Deployment Example at IXP

We consider the VIF IXP has a generic architecture that includes a layer-2 switching fabric, a route server (which is not highlighted in our paper), and a logical central controller for software-defined switches [36], [37]. Figure 10 illustrates a deployment example of VIF at an IXP. The filtering IXP sets up one or more commodity servers with SGX support.



(a) Vulnerable DNS resolvers (b) Mirai botnets

Fig. 11: The ratio of attack sources that are handled by the VIF filters for the two attack source data. (*): Top-n IXPs denote the n largest IXPs in each of the five regions, see Table III in Appendix H.

When a victim network is under DDoS attack, it contacts the controller of the VIF IXP via an out-of-band channel. As suggested in [63], the victim network can easily authenticate to the IXP via Resource Public Key Infrastructure (RPKI) [46]. The victim network asks the filtering IXP to create one or more SGX filters and audits it by receiving the validation attestation report(s). After being convinced that the filters have been set up properly (i.e., the remote attestation is successful), the victim network establishes a secure channel with the enclaves (e.g., TLS channels) and submits the filtering rules. The load balancing algorithm at the IXP controller receives the rules from the filter parallelization (see Section IV-B) and accordingly controls the switches to distribute traffic destined to the victim network to the enclaved filters. The VIF IXP eventually learns and analyzes all the rules in this step. Finally, the enclave filters perform packet filtering based on the submitted rules and forward the allowed traffic to the victim network.

C. Effectiveness of VIF at IXPs against DDoS Attacks

We analyze how much DDoS attack traffic can be filtered by VIF at IXPs with two real attack source data: 3 million vulnerable open DNS resolver IP addresses [79] and 250 thousand Mirai bot IP addresses [52].

Simulation setup. In our inter-domain routing simulation, we use the CAIDA Internet measurement data with the inferred AS business relationship [7] and the peering membership of world-wide IXPs [11]. We randomly choose 1,000 Tier-3 ISPs as the DDoS victims and consider that each victim receives attack traffic from all the attack sources (e.g., open resolvers and bots) in each case. To determine a traffic forwarding path between autonomous systems (ASes), we assume that each of them applies the following widely adopted BGP routing policies in order [29], [31]: (1) the AS prefers customer links over peer links and peer links over provider links; (2) the AS prefers the shortest AS-path length route; and (3) if multiple best paths exist, the AS uses the AS numbers to break the tie.

We assume that the victim network establishes VIF sessions with several largest IXPs (e.g., the biggest IXPs in each of the five regions, as shown in Table III, see Appendix H). We compute the ratio of flows from the attack IP addresses to the victim network that are handled by at least one of the established VIF filters at the selected IXPs. A traffic flow is said to be transited at an IXP if it traverses along an AS-path that include two consecutive ASes that are the members of the IXP.

Results. Figure 11 shows how many attack flows can be effectively handled by the in-network filters if installed in some large IXPs. The box-and-whisker plots show the distribution of the ratio of handled attack IPs when Top 1–5 biggest IXPs in the five regions (thus, 5–25 IXPs globally in total) perform in-network filtering service for DDoS defense. In each plot, the solid lines represent the first and the fourth quartile of the data set and the ends of the whisker indicate the 5th- and 95th-percentiles. Also, the red band inside the box represents the median.

Even when a single IXP in each region (thus, total five IXPs worldwide) adopts the VIF filters, the majority of both attack sources (e.g., vulnerable resolvers, botnet) are handled by the VIF IXPs. Approximately, 60% of attack mitigation is expected for the median cases, and 70-80% mitigation can be achieved for the top quarter cases. As more IXPs adopt the VIF filters, even more effective mitigation is achieved. Particularly, Top-5 IXPs per these regions appear to be sufficient enough to offer more than 75% attack mitigation for the median cases, and 80-90% of attack mitigation for the upper quarter cases.

D. Deployment Cost Analysis

Let us provide a ballpark estimate of the cost of deploying VIF at an IXP to handle 500 Gb/s of traffic. Note that the 500 Gb/s filter capacity at a *single* IXP appears to be sufficient because the attack volume at each IXP can be much lower than the aggregated volume at the victim network. For instance, it would require only a few VIF IXPs with similar capabilities to mitigate the biggest DDoS attack ever recorded with 1.7 Tb/s attack traffic [78].

Our experiment in Section V-B shows that a near full line-rate performance of 10 Gb/s per server with four SGX cores is easily achieved. Thus, to handle 500 Gb/s attack traffic, an IXP needs to invest in 50 modest SGX-supporting commodity servers, which would require only one or two server racks. With a commodity server cost is approximately US$ 2,000, the filtering IXP only needs to spend for one-time investment for US$ 100K to offer an extremely large defense capability of 500 Gb/s. The capital expenditure can be borne by the member ASes (hundreds or thousands) and/or can be amortized by the service fees if the filtering service is economically compensated by the payment from the victims [63]. A rigorous economic analysis of VIF operations in IXPs is out of the scope of this paper and is left for future work.

VII. Related Work

Network DDoS attacks and defenses have been extensively studied in the last 2–3 decades [54]. Here, we summarize a few categories of DDoS defenses and related projects.

A. In-network Filtering

The idea of in-network filtering has been the core idea of many DDoS mitigation proposals. There are two prominent approaches to implementation: dynamic filtering and capability-based approaches. Dynamic filtering suggests that the destination ISP requests the ISPs on the forwarding paths to install filter rules at the time of attack, for instance as proposed in Pushback [50], D-WARD [53], AITF [4], StopIt [48]. Capability-based approaches embed capabilities in the packet flows, which can be controlled by the destination

hosts to authorize flows in upstream, as proposed in SIFF [81], TVA [82], and Portcullis [57].

Closest to our work is the recent SENSS defense architecture by Ramanathan et al. [63]. SENSS proposes to install DDoS-victim submitted filters at a small number of major ISPs. Ramanathan et al. show that in-network filtering at only four major ISPs in the US would have stopped the Dyn attack happened in 2016 [84]. SENSS also suggests an automated payment channel between a DDoS victim and a filtering ISP so that the ISP can get compensation for the extra filtering tasks. Although the idea is solid and evaluation is promising, the SENSS proposal lacks the filtering verifiability and thus allows several undetectable misbehaviors of the filtering ISPs.[9]

Unlike previous in-line filtering proposals, our system VIF focuses on the highly desired but yet-unaddressed security property for in-network filtering system, i.e., the verifiable filter, and demonstrates its feasibility and scalability.

B. Secure Network Fault Localization

Existing secure network fault localization proposals [3], [8], [85] aim to identify the faulty network or link that causes undesirable packet drops in the Internet. When in-network filtering is deployed in transit networks, however, the network fault localization tools would become less useful. They can detect that a certain transit network drops packets; yet, the packet drops may, in fact, be the legitimate operation of DDoS traffic filtering requested by a remote DDoS victim.

Our VIF system enables the neighboring ASes and DDoS victim networks to distinguish whether certain packet drops are due to the DDoS defense or not. Moreover, it can verify the correct execution of the DDoS filters.

C. Network Function Virtualization with Trusted Hardware

We categorize some of them:

- **Middleboxes:** Various network middleboxes have been tested with the SGX capability. TLS middleboxes [39], [56] demonstrate that an SGX-protected middlebox can handle thousands of TLS sessions without violating their end-to-end encryption. ShieldBox [76] and Trusted Click [19] demonstrate that SGX can protect the Click modular router to implement various trusted network functions. S-NFV [67] also discusses general policy, data privacy issues of network functions. LightBox [25] demonstrates the line-rate performance for simple secure packet processing functions. Snort-SGX [45] also demonstrates the line-rate performance of Snort 3 along with a DPDK network layer. SafeBricks [59] implements a highly modularized and safe network function architecture in the Intel SGX platform.

  Our main contribution is not merely an integration of a simple flow filter function and an SGX architecture but more on addressing scalability and filter-rule violations at network layer that are specific to verifiable in-network filtering defense systems.
- **Privacy-preserving systems:** Several systems demonstrate that SGX-based network functions can improve the privacy of anonymity systems: SGX-protected Tor nodes [43], [70],
- **Inter-domain routing:** Also, several secure inter-domain routing applications have been proposed to leverage the security guarantees of Intel SGX [15], [44].
- **Verifiable accounting:** There also have been some prototype systems that enable the outsource network functions to securely measure the amount of resource used for the requested tasks (e.g., [75] in an SGX platform, [13] in a TPM platform).

With VIF we investigate a unique design point of auditable traffic filters. Particularly, our contribution of VIF is in the design of TEE based auditable filters that can handle DDoS attacks with an increase in the number of attack flows and the total bandwidth.

D. Cloud-based DDoS Mitigations

The predominant DDoS defense in practice today is an overlay-based filtering approach, such as cloud-based scrubbing services, that performs outsourced filtering in a third-party network on behalf of the DDoS victims (e.g., AWS-Shield [2], Radware DefensePro [62]). Overlay-based filtering approaches are popular particularly because they require *no* changes to the current Internet architecture. Recent works have proposed advances in such overlay filtering using middleboxes [30], [49] and proposals have discussed large-scale filtering locally at ISPs [26]. However, end users are not satisfied with the status quo. Reports suggest that relying on third-party providers centralizes the DDoS marketplace [30]; costs for small and medium-size victims are high and services are left to the discretion of large service providers [30], [80].

Unlike these proposals, VIF does not rely on the cloud or the local victim network's capability but directly establishes filtering rules at the transit networks.

VIII. Conclusion

In-network filtering has numerous known advantages over other proposed DDoS defenses; yet, it enables a potential malicious transit network to execute arbitrary filtering policies with impunity because of the lack of filtering verifiability. Our proof of concept VIF system demonstrates that verifiable in-network filtering is indeed possible with a practical hardware root of trust support. We hope that our study renews discussion on the deployment of in-network filtering in the IXPs and encourages more sophisticated yet auditable filter designs, such as stateful firewalls.

Acknowledgments

We thank the anonymous reviewers of this paper for their helpful feedback. We also thank Dmitrii Kuvaiskii, Virgil Gligor, and Hsu-Chun Hsiao for useful feedback on an early version of the paper. We thank Jun Seung You for his assistance in testing and publishing our open source codes. This research is supported in part by the National Research Foundation, Prime Ministers Office, Singapore under its National Cybersecurity R&D Program (TSUNAMi project, Award No. NRF2014NCR-NCR001-21). This research was also partially supported by a grant from Singapore Ministry of Education Academic Research Fund Tier-1 (R-252-000-624-133) and was supported in part by US NSF under award CNS-1565343.

[9] Ramanathan et al. [63] sketch a reputation-based mitigation, which can be used together with our VIF proposal.

REFERENCES

[1] B. Ager, N. Chatzis, A. Feldmann, N. Sarrar, S. Uhlig, and W. Willinger, "Anatomy of a large European IXP," *Proc. ACM SIGCOMM CCR*, 2012.

[2] "Amazon AWS Shield," https://aws.amazon.com/shield/, 2019.

[3] K. Argyraki, P. Maniatis, and A. Singla, "Verifiable network-performance measurements," in *Proc. ACM Co-NEXT*, 2010.

[4] K. J. Argyraki and D. R. Cheriton, "Active Internet Traffic Filtering: Real-Time Response to Denial-of-Service Attacks," in *Proc. USENIX ATC*, 2005.

[5] A. ARM, "Security technology building a secure system using trustzone technology (white paper)," *ARM Limited*, 2009.

[6] S. Arnautov, B. Trach, F. Gregor, T. Knauth, A. Martin, C. Priebe, J. Lind, D. Muthukumaran, D. O'Keeffe, M. Stillwell *et al.*, "SCONE: Secure Linux Containers with Intel SGX," in *Proc. OSDI*, 2016.

[7] "AS Relationships by CAIDA," http://www.caida.org/data/as-relationships/, 2019.

[8] C. Basescu, Y.-H. Lin, H. Zhang, and A. Perrig, "High-speed inter-domain fault localization," in *Proc. IEEE S&P*, 2016.

[9] A. Baumann, M. Peinado, and G. Hunt, "Shielding Applications from an Untrusted Cloud with Haven," in *Proc. OSDI*, 2015.

[10] P. Bright, "Can a ddos break the internet? sure... just not all of it," *Ars Technica*, 2013.

[11] "CAIDA Internet eXchange Points (IXPs) Dataset," https://www.caida.org/data/ixps/, 2019.

[12] N. Chatzis, G. Smaragdakis, A. Feldmann, and W. Willinger, "There is more to IXPs than meets the eye," in *Proc. ACM SIGCOMM*, 2013.

[13] C. Chen, P. Maniatis, A. Perrig, A. Vasudevan, and V. Sekar, "Towards verifiable resource accounting for outsourced computation," in *ACM SIGPLAN Notices*, 2013.

[14] S. Chen, X. Zhang, M. K. Reiter, and Y. Zhang, "Detecting privileged side-channel attacks in shielded execution with Déjá Vu," in *Proc. ACM AsiaCCS*, 2017.

[15] M. Chiesa, R. di Lallo, G. Lospoto, H. Mostafaei, M. Rimondini, and G. D. Battista, "PrIXP: Preserving the privacy of routing policies at Internet eXchange Points," in *Proc. IFIP/IEEE IM*, 2017.

[16] G. Cormode and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications," *Journal of Algorithms*, 2005.

[17] V. Costan and S. Devadas, "Intel SGX Explained," *IACR Cryptology ePrint Archive*, 2016.

[18] V. Costan, I. A. Lebedev, and S. Devadas, "Sanctum: Minimal Hardware Extensions for Strong Software Isolation," in *Proc. USENIX Security*, 2016.

[19] M. Coughlin, E. Keller, and E. Wustrow, "Trusted Click: Overcoming Security Issues of NFV in the Cloud," in *Proc. SDN-NFVSec*, 2017.

[20] I. I. Cplex, "V12. 1: Users Manual for CPLEX," *International Business Machines Corporation*, 2009.

[21] "DDoS Mon: Insight into Global DDoS Threat Landscape," https://ddosmon.net/insight/, 2019.

[22] C. Dietzel, A. Feldmann, and T. King, "Blackholing at IXPs: On the effectiveness of DDoS mitigation in the wild," in *Proc. PAM*, 2016.

[23] C. Dietzel, G. Smaragdakis, M. Wichtlhuber, and A. Feldmann, "Stellar: network attack mitigation using advanced blackholing," in *Proc. ACM CoNEXT*, 2018.

[24] "DPDK Pktgen," http://dpdk.org/browse/apps/pktgen-dpdk/refs/, 2019.

[25] H. Duan, X. Yuan, and C. Wang, "LightBox: Full-stack Protected Stateful Middlebox at Lightning Speed," *arXiv preprint arXiv:1706.06261v2*, 2018.

[26] S. K. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey, "Bohatei: Flexible and Elastic DDoS Defense," in *Proc. USENIX Security*, 2015.

[27] B. Fung, "What Europe can teach us about keeping the Internet open and free." in *The Washington Post*, Sept 20, 2013.

[28] R. Gandhi, H. H. Liu, Y. C. Hu, G. Lu, J. Padhye, L. Yuan, and M. Zhang, "Duet: Cloud scale load balancing with hardware and software," *ACM SIGCOMM CCR*, 2015.

[29] L. Gao, "On inferring autonomous system relationships in the Internet," *IEEE/ACM TON*, 2001.

[30] Y. Gilad, A. Herzberg, M. Sudkovitch, and M. Goberman, "CDN-on-Demand: An affordable DDoS Defense via Untrusted Clouds," in *Proc. NDSS*, 2016.

[31] P. Gill, M. Schapira, and S. Goldberg, "A survey of interdomain routing policies," *ACM SIGCOMM CCR*, 2013.

[32] D. Goltzsche, S. Rüsch, M. Nieke, S. Vaucher, N. Weichbrodt, V. Schiavoni, P.-L. Aublin, P. Costa, C. Fetzer, P. Felber, P. Pietzuch, and R. Kapitza, "EndBox: Scalable middlebox functions using client-side trusted execution," in *Proc. IEEE DSN*, 2018.

[33] J. Götzfried, M. Eckert, S. Schinzel, and T. Müller, "Cache attacks on Intel SGX," in *ACM EuroSec*, 2017.

[34] D. Gruss, J. Lettner, F. Schuster, O. Ohrimenko, I. Haller, and M. Costa, "Strong and efficient cache side-channel protection using hardware transactional memory," in *USENIX Security*, 2017.

[35] A. Gupta, N. Feamster, and L. Vanbever, "Authorizing network control at software defined Internet exchange points," in *Proc. SOSR*, 2016.

[36] A. Gupta, R. MacDavid, R. Birkner, M. Canini, N. Feamster, J. Rexford, and L. Vanbever, "An Industrial-Scale Software Defined Internet Exchange Point," in *Proc. NSDI*, 2016.

[37] A. Gupta, L. Vanbever, M. Shahbaz, S. P. Donovan, B. Schlinker, N. Feamster, J. Rexford, S. Shenker, R. Clark, and E. Katz-Bassett, "SDX: A software defined Internet exchange," *ACM SIGCOMM CCR*, 2015.

[38] M. Hähnel, W. Cui, and M. Peinado, "High-resolution side channels for untrusted operating systems," in *USENIX ATC*, 2017.

[39] J. Han, S. Kim, J. Ha, and D. Han, "SGX-Box: Enabling Visibility on Encrypted Traffic using a Secure Middlebox Module," in *Proc. ACM APNet*, 2017.

[40] "Intel Linux-SGX: `sgx_get_trusted_time`," https://github.com/intel/linux-sgx/issues/161, 2019.

[41] S. Johnson, V. Scarlata, C. Rozas, E. Brickell, and F. Mckeen, "Intel® Software Guard Extensions: EPID Provisioning and Attestation Services," 2016.

[42] E. Katz-Bassett, C. Scott, D. R. Choffnes, Í. Cunha, V. Valancius, N. Feamster, H. V. Madhyastha, T. Anderson, and A. Krishnamurthy, "LIFEGUARD: Practical repair of persistent route failures," in *Proc. ACM SIGCOMM*, 2012.

[43] S. M. Kim, J. Han, J. Ha, T. Kim, and D. Han, "Enhancing security and privacy of tor's ecosystem by using trusted execution environments." in *NSDI*, 2017.

[44] S. Kim, Y. Shin, J. Ha, T. Kim, and D. Han, "A first step towards leveraging commodity trusted execution environments for network applications," in *Proc. ACM HotNets*, 2015.

[45] D. Kuvaiskii, S. Chakrabarti, and M. Vij, "Snort® Intrusion Detection System with Intel® Software Guard Extension (Intel® SGX)," *arXiv preprint arXiv:1802.00508*, 2018.

[46] M. Lepinski, R. Barnes, and S. Kent, "An Infrastructure to Support Secure Internet Routing," *RFC 6480 (Informational)*, 2012.

[47] F. Liu, Y. Yarom, Q. Ge, G. Heiser, and R. Lee, "Last-Level Cache Side-Channel Attacks are Practical," in *Proc. IEEE S&P*, 2015.

[48] X. Liu, X. Yang, and Y. Lu, "To filter or to authorize: Network-layer DoS defense against multimillion-node botnets," in *Proc. ACM SIGCOMM*, 2008.

[49] Z. Liu, H. Jin, Y.-C. Hu, and M. Bailey, "MiddlePolice: Toward Enforcing Destination-Defined Policies in the Middle of the Internet," in *Proc. ACM CCS*, 2016.

[50] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Controlling high bandwidth aggregates in the network," *ACM SIGCOMM CCR*, 2002.

[51] F. McKeen, I. Alexandrovich, I. Anati, D. Caspi, S. Johnson, R. Leslie-Hurd, and C. Rozas, "Intel® software guard extensions (intel® sgx) support for dynamic memory management inside an enclave," in *Proc. of HASP*, 2016.

[52] "Mirai-like Botnet by Bad Packets Report," https://mirai.badpackets.net/, 2019.

[53] J. Mirkovic, G. Prier, and P. Reiher, "Attacking DDoS at the source," in *Proc. ICNP*, 2002.

[54] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM SIGCOMM CCR*, 2004.

[55] C. Morrow and R. Dobbins, "DDoS Open Threat Signaling (DOTS) Working Group: Operational Requirements," *Proc. IETF 93 Prague*, 2015.

[56] D. Naylor, R. Li, C. Gkantsidis, T. Karagiannis, and P. Steenkiste, "And Then There Were More: Secure Communication for More Than Two Parties," in *Proc. ACM CoNEXT*, 2017.

[57] B. Parno, D. Wendlandt, E. Shi, A. Perrig, B. Maggs, and Y.-C. Hu, "Portcullis: protecting connection setup from denial-of-capability attacks," *Proc. ACM SIGCOMM*, 2007.

[58] P. Patel, D. Bansal, L. Yuan, A. Murthy, A. Greenberg, D. A. Maltz, R. Kern, H. Kumar, M. Zikos, H. Wu *et al.*, "Ananta: Cloud scale load balancing," *ACM SIGCOMM CCR*, 2013.

[59] R. Poddar, C. Lan, R. A. Popa, and S. Ratnasamy, "Safebricks: Shielding network functions in the cloud," in *USENIX NSDI*, 2018.

[60] N. Porter, J. Garms, and S. Simakov, "Introducing Asylo: an open-source framework for confidential computing," 2018.

[61] Z. Pospichal, "New generation of DDoS mitigation in NIX.CZ," in *RIPE 74*, 2017.

[62] "Radware's DefensePro DDoS Protection," https://www.radware.com/products/defensepro/, 2019.

[63] S. Ramanathan, J. Mirkovic, M. Yu, and Y. Zhang, "SENSS Against Volumetric DDoS Attacks," in *Proc. ACSAC*, 2018.

[64] C. Rossow, "Amplification Hell: Revisiting Network Protocols for DDoS Abuse," in *Proc. NDSS*, 2014.

[65] M. Russinovich, "Introducing Azure confidential computing," in *Microsoft Azure Blog*, 2017.

[66] V. Scarlata, S. Johnson, J. Beaney, and P. Zmijewski, "Supporting Third Party Attestation for Intel® SGX with Intel® Data Center Attestation Primitives," 2018.

[67] M.-W. Shih, M. Kumar, T. Kim, and A. Gavrilovska, "S-NFV: Securing NFV states by using SGX," in *Proc. ACM SDN-NFV Security*, 2016.

[68] M.-W. Shih, S. Lee, T. Kim, and M. Peinado, "T-sgx: Eradicating controlled-channel attacks against enclave programs," in *Proc. NDSS*, 2017.

[69] S. Shinde, Z. L. Chua, V. Narayanan, and P. Saxena, "Preventing page faults from telling your secrets," in *Proc. AsiaCCS*, 2016.

[70] S. Shinde, D. Le Tien, S. Tople, and P. Saxena, "PANOPLY: Low-TCB Linux Applications With SGX Enclaves," in *Proc. NDSS*, 2017.

[71] J. M. Smith, K. Birkeland, and M. Schuchard, "An Internet-Scale Feasibility Study of BGP Poisoning as a Security Primitive," *arXiv preprint arXiv:1811.03716*, 2018.

[72] J. M. Smith and M. Schuchard, "Routing Around Congestion: Defeating DDoS Attacks and Adverse Network Conditions via Reactive BGP Routing," in *Proc. IEEE S&P*, 2018.

[73] D. Song, "Towards An Open-Source, Formally-Verified Secure Enclave," in *Workshop on Inter-Disciplinary Research Challenges in Computer Systems*, 2018.

[74] D. Storm, "Biggest DDoS attack in history slows Internet, breaks record at 300 Gbps," in *ComputerWorld*, 2013.

[75] S. Tople, S. Park, M. S. Kang, and P. Saxena, "VeriCount: Verifiable Resource Accounting Using Hardware and Software Isolation," in *ACNS*, 2018.

[76] B. Trach, A. Krohmer, F. Gregor, S. Arnautov, P. Bhatotia, and C. Fetzer, "Shieldbox: Secure middleboxes using shielded execution," in *Proc. ACM SOSR*, 2018.

[77] M. Tran, M. S. Kang, H.-C. Hsiao, W.-H. Chiang, S.-P. Tung, and Y.-S. Wang, "On the Feasibility of Rerouting-based DDoS Defenses," in *Proc. IEEE S&P*, 2019.

[78] L. Tung, "New world record DDoS attack hits 1.7Tbps days after landmark GitHub outage," in *ZDNet*, 2018.

[79] D. Wessels and A. Mohaisen, "Open Resolvers in COM/NET Resolution," *DNS-OARC Spring Workshop*, 2014.

[80] N. Woolf, "DDoS attack that disrupted internet was largest of its kind in history, experts say," in *ComputerWorld*, 2016.

[81] A. Yaar, A. Perrig, and D. Song, "SIFF: A stateless internet flow filter to mitigate DDoS flooding attacks," in *Proc. IEEE S&P*, 2004.

[82] X. Yang, D. Wetherall, and T. Anderson, "A DoS-limiting network architecture," in *Proc. ACM SIGCOM*, 2005.

[83] S. Yegulalp, "Level 3 accuses Comcast, other ISPs of 'deliberately harming' broadband service," in *InfoWorld*, 2014.

[84] K. York, "Dyn statement on 10/21/2016 DDoS attack," *Dyn Blog*, 2016.

[85] X. Zhang, C. Lan, and A. Perrig, "Secure and scalable fault localization under dynamic traffic patterns," in *Proc. IEEE S&P*, 2012.

APPENDIX A
FLOW-AWARE FILTER DESIGN

We consider deterministic and non-deterministic filter rule types:

- A *deterministic filter rule* defines a static {`ALLOW`, `DROP`} decision for a specified flow; and
- A *non-deterministic filter rule* expresses only the static probability distribution ($P_{\texttt{ALLOW}}$, $P_{\texttt{DROP}}$, where $P_{\texttt{ALLOW}} + P_{\texttt{DROP}} = 1$) for a specified flow and the final filter decision for each exact flows is made by the VIF filter. We guarantee the connection-preserving property in VIF filters so that all the packets in a TCP/UDP flow are allowed or dropped together.

The execution of non-deterministic rules with the *connection-preserving property* can be implemented in two different ways:

- *Hash-based filtering.* For each incoming packet p, we compute the cryptographic hash (e.g., SHA-256) of its five-tuple bits and the enclave's secrecy to make filtering decision based on the given probability distribution. For example, packet p is allowed if $H(\texttt{five-tuple-bits}||secrecy) < (2^{256} - 1) \times p_{\texttt{ALLOW}}$ with $H(\cdot)$ =SHA-256; and
- *Exact-match rule filtering.* For each TCP/UDP connection, the filter installs an exact-match rule with a filtering decision randomly chosen based on the given probability distribution.

Note that the two design points have different advantages and disadvantages. The hash-based filtering has a smaller memory footprint for lookup table but it incurs per-packet additional latency for cryptographic hash operations. In contrast, the exact-match filter design tends to have shorter per-packet processing time since it executes only one lookup but it requires a larger memory footprint for lookup tables and adds latency for frequent lookup table updates. We propose a *hybrid* design where hash-based filtering is performed for new flows until these new flows are installed with exact-match rules at every rule update period (e.g., 5–40 seconds).

APPENDIX B
DYNAMIC TEST OF INTERMEDIATE ASES

A victim network finds and avoids suspicious ASes that drop the VIF-allowed packets. Particularly, we utilize the well-known BGP *poisoning-based* inbound rerouting techniques (e.g., LIFEGUARD [42] and Nyx [72]) to reroute (or detour) inbound traffic and avoid traversing any intermediate ASes for a short period of time (e.g., a few tens of seconds). These BGP poisoning-based rerouting technologies do *not* require any inter-AS coordination and thus the victim network can independently test if any intermediate ASes drop packets (even without the VIF IXP's agreement).

$$\text{Minimize} \quad z$$

$$s.t. \qquad \forall p,q : z \geq \alpha\big(u\sum_{i=1}^{k} y_{i,p} + v\big) + \sum_{i=1}^{k} x_{i,q}y_{i,q} \tag{3}$$

$$\forall i : u \cdot \sum_{j=1}^{n} y_{i,j} + v \leq M \tag{4}$$

$$\forall j : \sum_{i=1}^{k} x_{i,j}y_{i,j} \leq G \tag{5}$$

$$\forall i : \sum_{j=1}^{n} x_{i,j} = b_i \tag{6}$$

$$\forall x_{i,j}, y_{i,j} : (1 - y_{i,j})x_{i,j} = 0 \tag{7}$$

$$\forall x_{i,j} \geq 0 \quad \text{and} \quad \forall y_{i,j} \in \{0,1\} \tag{8}$$

Fig. 12: ILP formulation for the optimal rule distribution.

If the misbehavior of an intermediate AS is detected by the victim network, then the misbehaving AS can be avoided for the extended period of time (at least during the VIF session) for auditable filtering. Or, if the victim network continuously witnesses that VIF-allowed packets are dropped continuously when dynamically changing the inbound routes, it may conclude that the VIF IXP itself has been misbehaving. The victim network can then discontinue the VIF contract with the VIF IXP at its discretion.

Note that we do not consider extremely adverse network adversaries, such as dropping all the packets between the VIF IXP and the victim network, which cannot be handled properly by any possible defenses in the current Internet architecture.

APPENDIX C
ILP FORMULATION FOR MULTI-ENCLAVE OPTIMIZATION

Our goal is to fully utilize the available resource on n enclaves in terms of the bandwidth and memory, without triggering the performance degradation. Also, the load on each enclave should be balanced, in order to reduce the chance of any enclave getting closer to the limit of G or M, see Figure 12. Hence, the maximum C_j and maximum I_j should all be as small as possible, as shown in Equation 3. Note that a constant coefficient α is used to balance two maximums in the sum. Because of the capacity limit of a single enclaved filter, any filter should have less than M memory consumption (Equation 4) and less than G bandwidth load (Equation 5). Since b_i is distributed to multiple filters, so their allocated bandwidth with respect to rule i should sum up to the value of b_i (Equation 6). Two decision variables are not independent since when $y_{i,j} = 0$, $x_{i,j}$ should never be a positive value. (Equation 7).

APPENDIX D
GREEDY ALGORITHM FOR SCALABLE FILTER DESIGN

The pseudocode in Algorithm 1 summarizes the greedy algorithm we use for the filter rule distribution problem for the scalable VIF filter design in Section IV-B.

Algorithm 1 Greedy algorithm for rule distribution and bandwidth allocation

procedure GREEDYSOLVER($b_1, b_2, ...b_k$, M, G, λ, u, v)
  $B \leftarrow \{b_1, b_2, ..., b_k\}$, $g \leftarrow \frac{1}{n}\sum_{i=1}^{k} b_i$, $h \leftarrow \frac{k}{n}$
  while $g \leq G$ **and** $h \leq (M - v)/u$ **do**
    $X \leftarrow$ ASSIGNBANDWIDTH(B, h, g, n)
    if $X \neq \emptyset$ **then**
      return X
    end if
    $g \leftarrow g + \Delta g$
    if $g > G$ **then**
      $h \leftarrow h + \Delta h$, $g \leftarrow \frac{1}{n}\sum_{i=1}^{k} b_i$
    end if
  end while
end procedure
procedure ASSIGNBANDWIDTH(B, h, g, n)
  $X \leftarrow \emptyset$, $j \leftarrow 1$
  while $B \neq \emptyset$ **and** $j \leq n$ **do**
    $r \leftarrow g$, $c \leftarrow 0$ ▷ c: remaining bandwidth for filter j; f: rule counter
    while $B \neq \emptyset$ **and** $c \leq h$ **do**
      $b_i \leftarrow$ PopMin(B)
      if $b_i < r$ **and** $j + 1 \leq h$ **then**
        $x_{i,j} \leftarrow b_i$, $X \leftarrow X \cup \{x_{i,j}\}$, $c \leftarrow c + 1$, $r \leftarrow r - b_i$
        continue
      end if
      $B \leftarrow B \cup \{b_i\}$, $b_i \leftarrow$ PopMax(B)
      if $b_i \leq r$ **then**
        $x_{i,j} \leftarrow b_i$, $X \leftarrow X \cup \{x_{i,j}\}$, $c \leftarrow c + 1$, $j \leftarrow j + 1$
      else
        $x_{i,j} \leftarrow r$, $X \leftarrow X \cup \{x_{i,j}\}$, $b_i \leftarrow b_i - r$, $B \leftarrow B \cup \{b_i\}$
      end if
      break
    end while
  end while
  if $B = \emptyset$ **then**
    return $\emptyset$
  end if
  return X
end procedure

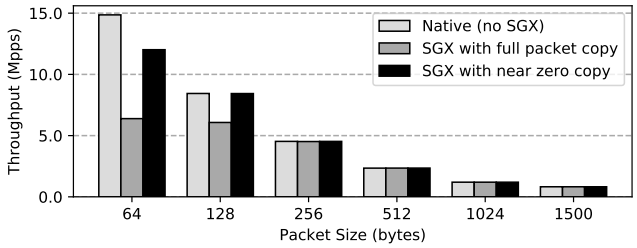


Fig. 13: Throughput performance in packet-per-second for varying packet sizes and 3,000 rules with three implementation versions: (1) Native (no SGX), (2) SGX with full packet copy, and (3) SGX with near zero copy.

TABLE II: Overhead of filter rule batch insertion to a multi-bit trie lookup table.

Number of rules in a batch	1	10	100	1000
Insert time (millisecond)	50	52	53	75

APPENDIX E
ADDITIONAL EVALUATION OF THROUGHPUT PERFORMANCE OF VIF

Figure 13 shows the throughput evaluation in packet per second metric. Taking a closer look at the SGX with full packet copy implementation, we notice that the maximum packet processing rate is capped at roughly 6 Mpps, which suggests the inherent capacity limit of the full packet-copy operations. Unlike the full packet copy version, the near zero-copy version shows no such throughput cap in terms of packet per second.

TABLE III: Top five IXPs in each of the five regions. Numbers inside the parentheses denote the member sizes of the IXPs.

Rank	Europe	North America	South America	Asia Pacific	Africa
1	AMS-IX (1660)	Equinix Ashburn (598)	IX.br São Paulo (2082)	Equinix Singapore (504)	NAPAfrica Johannesburg (506)
2	DE-CIX (1494)	Any2 (557)	PTT Porto Alegre (258)	Equinix Sydney (393)	NAPAfrica Cape Town (258)
3	LINX Juniper (755)	SIX (462)	PTT Rio de Janeiro (246)	Megaport Sydney (383)	JINX (180)
4	EPIX Katowice (732)	TorIX (426)	CABASE-BUE (183)	BBIX Tokyo (286)	NAPAfrica Durban (122)
5	LINX LON1 (697)	Equinix Chicago (384)	PTT Curitiba (140)	HKIX (281)	IXPN Lagos (69)

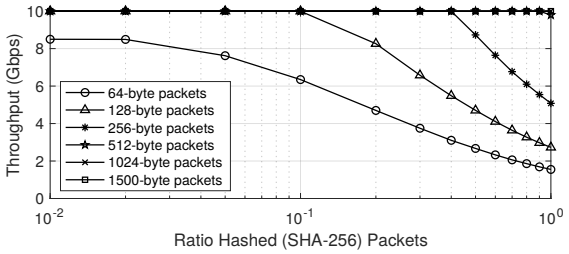


Fig. 14: Throughput performance of the maximum 10 Gbps VIF filter when a varying fraction of packets are hashed (i.e., SHA-256 is calculated for the 5-tuple bits).

APPENDIX F
CONNECTION-PRESERVING FILTERING PERFORMANCE OF VIF

As we discussed in Appendix A, the two mechanisms for connection preservation (i.e., hash-based filtering, exact-match rule based filtering) have different advantages and disadvantages. Thus, we present a *hybrid* design for practical operations. For any new flow that does not match any existing exact-match filter rules, the filter allows/drops based on the hash digest of the 5-tuple of the packets and queues this 5-tuple. At every filter rule update (e.g., every 5 seconds)[10], all the newly received flows since the last update are converted into exact-match rules and inserted to the lookup table. This hybrid design amortizes the cost of lookup table update by batch processing multiple newly observed flows at every update period. Also, it limits the per-packet latency increase due to hash operations since newly observed flows should be the minority in general. Indeed, our experiments on the performance overhead of the use of hash-based filtering with various packet sizes show no performance degradation, except with small packet size.

Figure 14 shows our experiment on the performance overhead of the use of hash-based filtering for varying fraction of incoming packets. Particularly, when the ratio of hashed packets is low (e.g., less than 10%), we observe no performance degradation in all packet sizes, except the smallest size (i.e., 64-byte) where up to 25% throughput degradation is measured. We argue that this performance degradation is easily acceptable because in general, the fraction of newly observed flows within a short period (e.g., 5 seconds) would be small. Moreover, the 64-byte performance degradation in Figure 14 must be the lower-bound result since it assumes that all the packets are 64 bytes.

Table II shows the benchmark on the time taken to insert the batched new exact-match rules to a multi-bit tri-based lookup table. Our test shows that the batch insertion of filter rules is quite efficient and incurs minimal performance overhead even for large batch size; e.g., only 75 milliseconds compared to the 5-second rule update period.

[10]The rule update period can be synchronized with that of the rule reconfiguration for scalable, multiple enclave operations; see Section IV-B.

APPENDIX G
REMOTE ATTESTATION PERFORMANCE

VIF performs remote attestation for each new enclave that the VIF IXP launches on its infrastructure. Since VIF is expected to operate under DDoS attacks, we want to ensure that the launching of multiple-enclaves on demand does not become the bottleneck for our deployment model. Here, we measure the total amount of time to complete an end-to-end remote attestation process for one enclave. In our micro-benchmark for remote attestation for the conservative performance tests, we set up the filter enclave and the destination on a cloud machine hosted in South Asia, and the IAS service hosted in Ashburn, Virginia, United States. For an enclave binary of size 1 MB, the platform takes 28.8 milliseconds and the total end-to-end latency of 3.04 seconds with a standard deviation of 9.2 milliseconds.

APPENDIX H
TOP REGIONAL IXPS

We use the IXP peering membership from CAIDA [11] to count the number of AS members of each IXP and summarize the top five IXPs in each of five regions (Europe, North America, South America, Asia Pacific and Africa) in Table III.