Practical Proactive DDoS-Attack Mitigation via Endpoint-Driven In-Network Traffic Control

Zhuotao Liu, Hao Jin, Yih-Chun Hu, and Michael Bailey

***Abstract*— Volumetric attacks, which overwhelm the bandwidth of a destination, are among the most common distributed denial-of-service (DDoS) attacks today. Despite considerable effort made by both research and industry, our recent interviews with over 100 potential DDoS victims in over 10 industry segments indicate that today's DDoS prevention is far from perfect. On one hand, few academical proposals have ever been deployed in the Internet; on the other hand, solutions offered by existing DDoS prevention vendors are not silver bullet to defend against the entire attack spectrum. Guided by such large-scale study of today's DDoS defense, in this paper, we present MiddlePolice, the first readily deployable and proactive DDoS prevention mechanism. We carefully architect MiddlePolice such that it requires no changes from both the Internet core and the network stack of clients, yielding instant deployability in the current Internet architecture. Further, relying on our novel capability feedback mechanism, MiddlePolice is able to enforce destination-driven traffic control so that it guarantees to deliver victim-desired traffic regardless of the attacker strategies. We implement a prototype of MiddlePolice and demonstrate its feasibility via extensive evaluations in the Internet, hardware testbed, and large-scale simulations.**

***Index Terms*— Network security, internet technology, middleboxes.**

I. Introduction

BECAUSE the Internet internally does not enforce any traffic control requirements, a number of attacks have been developed to overwhelm Internet end systems. The most significant of these attacks is the volumetric Distributed Denial-of-Service (DDoS) attack, representing over 65% of all DDoS attacks in 2015 [2]. In a volumetric DDoS, many attackers coordinate and send high-rate traffic to a victim, in an attempt to overwhelm the bottleneck links close to the victim. Typical Internet links use RED and drop-tail FIFO queuing disciplines, which provide nearly-equal loss rates to all traffic. Consequently, saturated links impose equal loss rates on attacking and legitimate traffic alike. While legitimate traffic tends to back off to avoid further congestion, attack traffic does not back off, so links saturated by a DDoS attack are effectively closed to legitimate traffic. Recent DDoS attacks include a 620 Gbps attack against Krebs' security blog [3] and a 1 Tbps attack against OVH [4], a French ISP.

Over the past few decades, both the industry and research have made considerable effort to address this problem. Academics proposed various approaches, ranging from filtering-based approaches [5]–[10], capability-based approaches [1], [11]–[13], overlay-based systems [14]–[16], systems based on future Internet architectures [17]–[19] and other approaches [20]–[22]. Meanwhile, many cloud security service providers (*e.g.,* Akamai, Cloudflare) have played an important role in practical DDoS prevention. These providers massively over-provision data centers for peak attack traffic loads and then share this capacity across many customers as needed. When under DDoS attack, victims use DNS or BGP to redirect their traffic to the provider rather than their own networks. These providers apply their proprietary techniques to scrub traffic, separating malicious from benign, and then re-inject the remaining traffic back into the network of their customers.

Although these academic and industrial solutions have been proposed for years, the research literature offers surprisingly few real-world studies about their performance and the current status of DDoS prevention. Thus, supported by the NSF I-Corps program, we initiated a large-scale interview with over 100 potential DDoS victims spread across more than ten industry segments to understand (i) their opinions about academic proposals, (ii) their current practice for DDoS prevention and (iii) their opinions about these common practices. Our study highlights multiple key observations, two of which are the indeployability of most academic proposals and the "band-aid" nature of the scrubbing services offered by these cloud security service providers.

Our study clearly establishes that any advanced DDoS prevention system must meet the following two key challenges. *(i) Instant Deployability.* Any practical DDoS prevention proposal must be readily deployable in the current Internet architecture. This is challenging because the current Internet contains over 60 000 independent Autonomous Systems (ASes), with varying levels of technological sophistication and cooperativeness. As a result, prior academic approaches (*e.g.,* SIFF [11], TVA [12], and NetFence [13]) that require secret key management and router upgrades across a large number of ASes face significant deployment hurdles. To address this challenge, we carefully architect our propose system MiddlePolice such that it requires no changes from both the Internet core and network stack of clients, yielding instant deployability. *(ii) Proactive Defense.* As we shall see, the filtering services offered by existing Cloud Security Service Providers (CSSPs) are not a silver bullet as the majority of potential DDoS victims that purchase services from CSSPs still experience

Manuscript received September 29, 2017; revised June 16, 2018; accepted June 30, 2018; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor W. Lou. Date of publication July 23, 2018; date of current version August 16, 2018. This work was supported by the National Science Foundation under Grants CNS-1717313, IIP-1758179, and CNS-0953600. Part of the material in this work appears in [1]. *(Corresponding author: Zhuotao Liu.)*

Z. Liu, Y.-C. Hu, and M. Bailey are with the Electrical and Computer Engineering Department, University of Illinois at Urbana–Champaign, Urbana, IL 61801 USA (e-mail: zliu48@illinois.edu; yihchun@illinois.edu; mdbailey@illinois.edu).

H. Jin is with Computer Science and Technology, Nanjing University, Nanjing 210008, China (e-mail: jinhaonju@gmail.com).

Digital Object Identifier 10.1109/TNET.2018.2854795

problems. The fact that even these CSSPs with decades of DDoS mitigation experience cannot develop perfect filtering techniques probably indicates that it is difficult to win the arms race with attackers simply via reactive filtering. Thus, we argue that an advanced DDoS mitigation should focus on delivering victim-desired traffic even without understanding characteristics of attacks. To this end, MiddlePolice is designed to enforce destination-driven traffic control for DDoS mitigation so as to minimize any possible disruption caused by the attacks.

To summarize, our key contributions are:

- A study of today's DDoS defense involving over 100 potential DDoS victims in more than 10 industry segments.
- A systematic analysis of our findings to clarify the design space of advanced DDoS prevention mechanisms.
- The design, implementation and evaluation of MiddlePolice, the first system offering readily deployable and proactive DDoS mitigation.

II. DDoS Defense Today

Before discussing our proposed system, we first present our study of current status of real-world DDoS attacks and defense. Supported by the NSF Innovation Corps program under grant IIP-1758179, we interviewed more than 100 security engineers/administrators from over ten industrial segments, including hosting companies, financial departments, online gaming providers, military contractors, government institutes, medical foundations, and existing DDoS prevention vendors. To the best of our knowledge, in the research community, this is the first comprehensive study of DDoS prevention from the perspective of security experts that are the first-line DDoS defenders. Our analysis highlights following key observations.

A. Deployment of Academic Proposals

Over the past decades, the research community have proposed various approaches, see discussion in § X. From the research perspective, most of these proposals are provably secure and ensure that a DDoS victim can suppress unwanted traffic. Unfortunately, over the past decades, few research proposals have ever been deployed in the Internet.

Such lack of real-world deployment is because potential DDoS victims (even large ones such as international financial departments) are unable to enforce the deployment of academic proposals in the Internet. For instance, some proposals require software/hardware upgrades from a large number of geographically distributed ASes. However, due to the lack of business relationship with these remote ASes, these victims are incapable of enforcing any deployment within these ASes. Among all our interviews with security administrators, few of them mentioned that they would consider academic proposals for their practical DDoS prevention.

B. The Market Giants

In practice, potential DDoS victims rely on DDoS prevention providers to keep them online. In our study, we interviewed ten market giants in this space, including cloud security-service providers that build massively-provisioned data centers to scrub attack traffic and vendors that sell on-premise hardware equipments designed for DDoS prevention. We make the following key observations.

1) Cloud Security-Service Providers: Cloud Security-Service Providers (CSSPs) play an important role for DDoS prevention: over 80 percent of our interviewed potential DDoS victims purchase services from CSSPs. Although their actual products could differ, these CSSPs typically work as follows: deploying geographically distributed *automatic systems* (referred to as sites), terminating customer traffic at these sites, filtering traffic using proprietary rules, and reinjecting the scrubbed traffic back to their customers. CSSPs indeed offer invaluable services to their customers. However, their defense is not a silver bullet. In particular, two caveats are worth mentioning. First, connection termination at CSSPs' sites is privacy invasive for some large organizations, such as government institutes and medical foundations. Second, their "secret sauce" filtering technique typically deals with large-yet-obvious attacks (*i.e.,* although traffic volumes are huge, these attacks are "trivial" to distinguish via static filters, such as port filtering). Quoting one administrator:

> *I believe automatic systems managed by vendors should only deal with large things (attacks). Getting into discussion of customer-specific filtering is a slippery slope.*

In §II-C, we discuss how these caveats might affect DDoS victims that use protection services from CSSPs.

2) On-Premise DDoS Prevention Hardware: Another type of vendor in DDoS prevention space is hardware manufacturers. Unlike CSSPs that build massive infrastructure, these vendors offer specialized hardware with built-in DDoS prevention capability, such as configurable filtering rules, traffic monitoring and advanced queuing. About 45% of our interviewed potential DDoS targets deploy on-premise hardware.

C. Potential DDoS Victims

We interviewed security engineers/administrators from potential DDoS victims that are fairly large organizations whose online presence is critical for their business. Examples include web hosting companies, online content providers, financial departments, medical foundations, government facilities and so on. We summarize our findings as follows.

1) Mixed Opinions Towards CSSPs: Over 80% of all interviewed potential DDoS victims purchase scrubbing services from one of these CSSPs. Their opinions towards these services are mixed. Some security administrators said that these providers do a "decent" job to prevent attacks. However, the majority of interviewed administrators claim that they experience attacks even after purchasing these services. Although these attacks are not large enough to knock them offline completely, they still result in various performance issues, including system instability, severe packet losses and even customer churn. Quoting one administrator:

> *We do understand their (CSSPs) filters are a band-aid. But these are common practice.*

This observation echoes our discussion in §II-B1. Since CSSPs mainly focus on dealing with large-yet-obvious attacks, any attack traffic that bypasses their filters will eventually reach end-customers. Although these attacks are "small" by a CSSP's definition, they are large enough to cause problems for those DDoS targets.

2) On-Demand Filtering From ISPs: About 20% of our interviewed victims (such as medical and goverment institutes) do not rely on CSSPs. The primary reason is privacy concerns: they cannot afford to allow a third party to have full access to their network connections. As a result, they typically deploy

on-premise devices to closely monitor network traffic, and once the traffic volume is above a pre-defined threshold, they work together with their ISPs to filter these offending flows.

D. Survey Summary

To recap, our survey covers over 100 potential DDoS victims in more than ten industry segments. The key observations are summarized as follows. (i) Since most of the academic proposals incur significant deployment overhead in the Internet, few of them have ever been deployed in the Internet. (ii) Current CSSPs that dominate the market mainly care about scrubbing large-yet-obvious attacks based on empirical filtering rules. (iii) Some potential DDoS victims are attacked continuously even if they purchase services from CSSPs. (iv) Some potential victims that cannot afford to allow CSSPs to terminate their network connections due to privacy concerns have to rely on their ISPs to block offending flows.

III. Design Goals and Assumptions

A. Design Space and Goals

Having understood the status of real-world DDoS defense, when designing MiddlePolice, we explicitly achieve two primary goals: being readily deployable in the current Internet and offering proactive mitigation even against sophisticated DDoS attacks. To achieve instant deployability, we carefully architect MiddlePolice such that it requires no changes from both the Internet core and the network stack of clients. Further, the fact that even these large CSSPs with decades of DDoS mitigation experience are unable to offer satisfactory defense indicates that probably it is difficult to win the arms race with attackers by only reactive filters. Rather, we argue that effective DDoS mitigation, to a large extent, is about delivering victim-desired traffic even without understanding the characteristics of attacks. Such *proactive* DDoS mitigation can minimize any potential disruption caused by attacks, regardless of how adversaries may adjust their strategies. In summary, to offer readily deployable and proactive DDoS mitigation, MiddlePolice enables the following three key properties.

Readily Deployable and Scalable: MiddlePolice is designed to be readily deployable in the Internet and sufficiently scalable to handle large scale attacks. *To be readily deployable, a system should only require deployment at the destination, and possibly at related parties on commercial terms.* The end-to-end principle of the Internet, combined with large numbers of end points, is what gives rise to its tremendous utility. Because of the diversity of administrative domains, including end points, edge-ASes, and small transit ASes, ASes have varying levels of technological sophistication and cooperativeness. However, some ASes can be expected to help with deployment; many ISPs already provide some sort of DDoS-protection services [23], so we can expect that such providers would be willing to deploy a protocol under commercially reasonable terms. We contrast this with prior capability-based work, which requires deployment at a large number of unrelated ASes in the Internet and client network stack modification, that violates the deployability model.

The goal of being deployable and scalable is the pushing reason that MiddlePolice is designed to be deployable in cloud infrastructure without changing the Internet core.

Destination-Driven Traffic Control: MiddlePolice is designed to provide the destination with fine-grained control over its network utilization during DDoS mitigation. Throughout the paper, we use "destination" and "victim" interchangeably. Existing CSSPs have not provided such functionality. Many previously proposed capability-based systems are likewise designed to work with a single scheduling policy. For instance, CRAFT [24] enforces per-flow fairness, Portcullis [25] and Mirage [21] enforce per-compute fairness, NetFence [13] enforces per-sender fairness, SIBRA [26] enforces per-steady-bandwidth fairness, and SpeakUp [20] enforces per-outbound-bandwidth fairness. If any of these mechanisms is ever deployed, a single policy will be enforced, forcing the victim to accept the choice made by the defense approach. However, no single fairness regime can satisfy all potential victims' requirements. Ideally, MiddlePolice should be able to support arbitrary victim-chosen traffic control policies. In addition to these fairness metrics, MiddlePolice can implement ideas such as ARROW's [27] special pass for critical traffic, and prioritized services for premium clients.

To the best of our knowledge, MiddlePolice is the first DDoS prevention mechanism that advocates destination-driven traffic control for DDoS mitigation. We argue that destination-driven traffic control is the key to offer proactive mitigation even against sophisticated DDoS attacks. There is a fundamental difference between destination-driven traffic control and any vendor or protocol defined traffic control. In particular, compared with vendors or network protocol designers, potential DDoS victims may have much more specific and complete knowledge base about their own desired traffic, for instance, through comprehensive traffic monitoring and analysis. As a result, the victim can make more rational forwarding decisions that are coherent with their business and applications. Therefore, regardless of how sophisticated a DDoS attack is, as long as MiddlePolice can effectively enforce victim-defined traffic control policies to forward victim-preferred traffic, the impact imposed by the DDoS attack on a victim is minimized.

Fixing the Bypass Vulnerability: Since MiddlePolice is designed to be deployed in cloud infrastructure, we need to further address an open vulnerability. In particular, existing cloud-based DDoS-prevention vendors rely on DNS or BGP to redirect the destination's traffic to their infrastructures. However, this model opens up the attack of infrastructure bypass. For example, a majority of cloud-protected web servers are subject to IP address exposure [28], [29]. Larger victims that SWIP their IP addresses may be unable to keep their IP addresses secret from a determined adversary. In such cases, the adversary can bypass the cloud infrastructure by routing traffic directly to the victims. MiddlePolice includes a readily deployable mechanism to address this vulnerability.

B. Adversary Model and Assumptions

Adversary Model: We consider a strong adversary owning large botnets that can launch strategic attacks and amplify its attack [30]. We assume the adversary is not on-path between any mbox and the victim, since otherwise it could drop all packets. Selecting routes without on-path adversaries is an orthogonal problem and is the subject of active research in next-generation Internet protocols (*e.g.,* SCION [17]).

Well-Connected: MiddlePolice is built on a distributed and replicable set of mboxes that are well-connected to the Internet backbone. We assume the Internet backbone has sufficient capacity and path redundancy to absorb large volumes of traffic, and DDoS attacks against the set of all mboxes can never be successful. This assumption is a standard assumption for cloud-based systems.

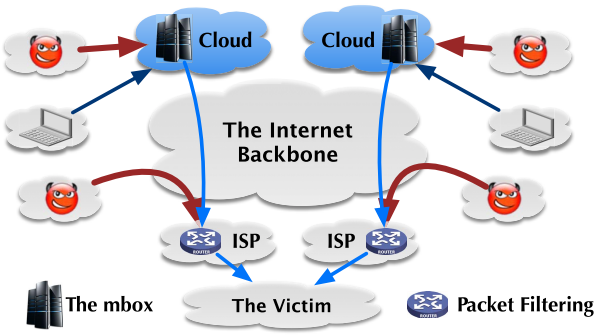

Fig. 1. The architecture of MiddlePolice. The mboxes police traffic to enforce victim-selected traffic control policies. The packet filtering can discard all traffic bypassing upstream mboxes.

Victim Cooperation: MiddlePolice's defense requires the victim's cooperation. If the victim can hide its IP addresses from attackers, it simply needs to remove a MiddlePolice-generated capability carried in each packet and return it back to the mboxes. The victim needs not to modify its layer-7 applications as the capability feedback mechanism is transparent to applications. If attackers can directly send or point traffic (*e.g.,* reflection) to the victim, the victim needs to block the bypassing traffic. MiddlePolice includes a packet filtering mechanism that is readily deployable on commodity Internet routers with negligible overhead.

Cross-Traffic Management: We assume that bottlenecks on the path from an mbox to the victim that is shared with other destinations are properly managed, such that cross-traffic targeted at another destination cannot cause unbounded losses of the victim's traffic. Generally, per-destination-AS traffic shaping (*e.g.,* weighted fair share) on these links will meet this requirement.

IV. System Overview

MiddlePolice's high-level architecture is illustrated in Figure 1. A MiddlePolice-protected victim redirects its traffic to the mboxes. Each mbox polices traversing traffic to enforce the traffic control policy chosen by the victim. The traffic policing relies on a feedback loop of MiddlePolice-generated capabilities to eliminate the deployment requirements on downstream paths. When the victim keeps its IP addresses secret, a single deploying mbox can secure the entire downstream path from the mbox to the victim.

For victims whose IP addresses are exposed, attackers can bypass the mboxes and direct attack traffic to the victim. The same vulnerability applies for current existing cloud-based DDoS-prevention systems. To address this problem, MiddlePolice designs a packet filtering mechanism relying on the ACL on commodity routers or switches to eliminate the traffic that does not traverse any mbox. As long as each bottleneck link is protected by an upstream filter, the bypass attack can be prevented.

V. Detailed Design of mboxes

MiddlePolice's traffic policing algorithm (i) probes the available downstream bandwidth from each mbox to the victim and (ii) allocates the bandwidth to senders to enforce the traffic control policies chosen by the victim.

Bandwidth Probe: The fundamental challenge of estimating downstream bandwidth is that MiddlePolice requires no deployment at downstream links. Such a challenge is two-sided: an overestimate will cause downstream flooding, rendering traffic policing useless, while an underestimate will waste downstream capacity, affecting networking performance.

To solve the overestimation problem, MiddlePolice relies on a *capability feedback* mechanism to make sources self-report how many packets they have successfully delivered to the victim. Specifically, upon a packet arrival, the mbox stamps an unforgeable capability in the packet. When the packet is delivered to the victim, MiddlePolice's capability handling module (CHM) deployed on the victim returns the carried capability back to the mbox. If the capability is not returned to the mbox after a sufficiently long time interval (compared with the RTT between the mbox and victim), the mbox will consider the packet lost. Thus, the feedback enables the mbox to infer a packet loss rate (hereinafter, LLR) for each sender. Then the mbox estimates the downstream capacity as the difference between the number of packets received from all senders and packets lost on the downstream path. As the estimation is based on the traffic volume *delivered* to the victim, this approach solves the overestimation problem.

However, the above technique does not overcome the underestimation problem. Specifically, since the traffic demand may be less than downstream capacity, simply using the volume of delivered traffic may cause underestimation. To prevent underestimation, the mbox categorizes packets from each sender as *privileged packets* and *best-effort* packets. Specifically, the mbox maintains a rate window $\mathcal{W}_R$ for each sender to determine the amount of privileged packets allowed for the sender in each period (hereinafter, *detection period*). $\mathcal{W}_R$ is computed based on the above downstream capacity estimation as well as victim-chosen policies. Packets sent beyond $\mathcal{W}_R$ are classified as best-effort packets. The mbox forwards all privileged packets to the victim, whereas the forwarding decisions for best-effort packets are subject to a short-term packet loss rate (hereinafter, SLR). The SLR reflects downstream packet loss rates (congestion) at a RTT granularity. That is, if the downstream is not congested upon an arrival of a best-effort packet, the mbox will forward the packet. Thus, even when the downstream capacity (and thus $\mathcal{W}_R$) is underestimated, the mbox can still further deliver packets as long as the downstream path is not congested.

Fairness Regimes: Each mbox allocates its bandwidth estimate amongst its senders based on the traffic control policies chosen by the victim. For policies enforcing global fairness among all senders, all mboxes sharing the same bottleneck need to share their local observations. We discuss co-bottleneck detection in § IX-C.

A. Information Table

The basis of MiddlePolice's traffic policing is an information table (*iTable*) maintained by each mbox. Each row of the *iTable* corresponds to a single sender. The contents of the *iTable* (and therefore its complexity and cost of maintaining the table) depend on the victim-selected traffic control policy; this section describes *iTable* elements needed for enforcing per-sender fairness, and § V-C5 extends the *iTable* to other traffic control policies. In § VII, we describe multiple mechanisms to filter source spoofing at the mbox, so this section ignores source spoofing.

TABLE I
FIELDS OF AN *iTable* ENTRY AND THEIR SIZES (BITS)

f	$\mathcal{T}_A$	$\mathcal{P}_{id}$	$\mathcal{N}_R$	$\mathcal{N}_D$	$\mathcal{W}_R$	$\mathcal{W}_V$	$\mathcal{L}_R$
64	32	16	32	32	32	128	64

Each sender s_i has one row in the *iTable*, identified by a unique identifier f. The table contents are illustrated in Table I. Other than f, the remaining fields are updated in each *detection period*. The timestamp $\mathcal{T}_A$ records the current detection period. The capability ID $\mathcal{P}_{id}$ is the maximum number of distinct capabilities generated for s_i. $\mathcal{N}_R$ stores the number of packets received from s_i. $\mathcal{N}_D$ indicates the number of best-effort packets dropped by the mbox. $\mathcal{W}_R$ determines the maximum number of privileged packets allowed for s_i. The verification window $\mathcal{W}_V$ is designed to compute s_i's packet loss rate, whereas $\mathcal{L}_R$ stores the LLR for s_i.

B. Capability Computation

For s_i, the mbox generates two types of capabilities: distinct capabilities and common capabilities. The CHM can use either capability to authenticate that the packet has traversed the mbox, though only distinct capabilities are used to infer downstream packet losses.

A distinct capability for s_i is computed as follows:

$$\mathcal{C} = IP_{MP} \mid\mid ts \mid\mid \mathcal{P}_{id} \mid\mid f \mid\mid \mathcal{T}_A \mid\mid MAC_{K_s}(IP_{MP} \mid\mid ts \mid\mid \mathcal{P}_{id} \mid\mid f \mid\mid \mathcal{T}_A), \quad (1)$$

where IP_{MP} is the IP address of the mbox issuing $\mathcal{C}$ and *ts* is the current timestamp (included to mitigate replay attack). The combination of $\mathcal{P}_{id}||f||\mathcal{T}_A$ ensures the uniqueness of $\mathcal{C}$. The MAC is computed based on a secret key K_s shared by all mboxes. The MAC is 128 bits, so the entire $\mathcal{C}$ consumes $\sim$300 bits. A common capability is defined as follows

$$\mathcal{C}_c = IP_{MP} \mid\mid ts \mid\mid MAC_{K_s}(IP_{MP} \mid\mid ts). \quad (2)$$

The design of capability incorporates a MAC to ensure that attackers without secure keys cannot generate valid capabilities, preventing capability abuse.

C. Traffic Policing Logic

1) Populating the iTable: We first describe how to populate the *iTable*. At time *ts*, the mbox receives the first packet from s_i. It creates an entry for s_i, with f computed based on s_i's source address, and initializes the remaining fields to zero. It then updates $\mathcal{T}_A$ to ts, increases both $\mathcal{N}_R$ and $\mathcal{P}_{id}$ by one to reflect the packet arrival and computes a capability using the updated $\mathcal{P}_{id}$ and $\mathcal{T}_A$.

New packet arrival from source s_i may trigger the mbox to start a new detection period for s_i. In particular, upon receiving a packet from s_i with arrival time $t_a - \mathcal{T}_A > \mathcal{D}_p$ ($\mathcal{D}_p$ is the length of the detection period), the mbox starts a new detection period for s_i by setting $\mathcal{T}_A = t_a$. The mbox also updates the remaining fields based on the traffic policing algorithm (as described in § V-C4). The algorithm depends on s_i's LLR and the mbox's SLR, the computation of which is described in the following two sections.

2) Inferring the LLR for Source s_i:

TABLE II
SYSTEM PARAMETERS

Symb.	Definition	Value
$\mathcal{D}_p$	The length of the detection period	$4s$
Th_{cap}	The upper bound of capability ID	128
Th_{rtt}	Maximum waiting time for cap. feedback	1s
Th_{slr}^{drop}	SLR thres. for dropping best-effort pkts	0.05
β	The weight of historical loss rates	0.8
Th_{lpass}	The threshold for calculating LLR	5
S_{slr}	The length limit of the *cTable*	100

Capability generation: For each packet from s_i, the mbox generates a distinct capability for the packet if (i) its arrival time $t_a - \mathcal{T}_A < \mathcal{D}_p - Th_{rtt}$, and (ii) the capability ID $\mathcal{P}_{id} < Th_{cap}$. The first constraint ensures that the mbox allows at least Th_{rtt} for each capability to be returned from the CHM. By setting Th_{rtt} well above the RTT from the mbox to the victim, any missing capabilities at the end of the current detection period correspond to lost packets. Table II lists the system parameters including Th_{rtt} and their suggested values. MiddlePolice's performance with different parameter settings is studied and analyzed in § IX-C. The second constraint Th_{cap} bounds the number of distinct capabilities issued for s_i in one detection period, and thus bounds the memory requirement. We set $Th_{cap} = 128$ to reduce the LLR sampling error while keeping memory overhead low.

Packets violating either of the two constraints, if any, will carry a common capability (Equation (2)), which is not returned by the CHM or used to learn s_i's LLR. However, it can be used as an authenticator that the packet has been accepted by one of the upstream mboxes.

Capability feedback verification: Let K_{th} denote the number of distinct capabilities the mbox generates for s_i, with capability ID ranging from $[1, K_{th}]$. Each time the mbox receives a returned capability, it checks the capability ID to determine which packet (carrying the received capability) has been received by the CHM. $\mathcal{W}_V$ represents a window with Th_{cap} bits. Initially all the bits are set to zero. When a capability with capability ID i is received, the mbox sets the ith bit in $\mathcal{W}_V$ to one. At the end of the current detection period, the zero bits in the first K_{th} bits of $\mathcal{W}_V$ indicate the losses of the corresponding packets. To avoid feedback reuse, the mbox only accepts feedback issued in the current period.

LLR computation: LLR in the kth detection period is computed at the end of the period, *i.e.,* the time when the mbox starts a new detection period for s_i. s_i's lost packets may contain downstream losses and best-effort packets dropped by the mbox ($\mathcal{N}_D$). The number of packets that s_i sent to downstream links is $\mathcal{N}_R - \mathcal{N}_D$, and the downstream packet loss rate is $\frac{V_0}{\mathcal{P}_{id}}$, where V_0 is the number of zero bits in the first $\mathcal{P}_{id}$ bits of $\mathcal{W}_V$. Thus, the estimated number of downstream packet losses is $\mathcal{N}_{loss}^{dstream} = (\mathcal{N}_R - \mathcal{N}_D)\frac{V_0}{\mathcal{P}_{id}}$. Then we have $LLR = (\mathcal{N}_{loss}^{dstream} + \mathcal{N}_D)/\mathcal{N}_R$.

Our strawman design is subject to statistical bias, and may have negative effects on TCP timeouts. In particular, assume one legitimate TCP source recovers from a timeout and sends one packet to probe the network condition. If the packet is dropped again, the source will enter a longer timeout. However, with the strawman design, the source would incorrectly have a 100% loss rate. Adding a low-pass filter can fix this

problem: if s_i's $\mathcal{N}_R$ in the current period is less than a small threshold Th_{lpass}, the mbox sets its LLR in the current period to zero. Attackers may exploit this design using on-off attacks [31]. In § V-C4, we explain how to handle such attacks. Formally, we write *LLR* as follows

$$LLR = \begin{cases} 0, & \text{if } \mathcal{N}_R < Th_{lpass} \\ \dfrac{\mathcal{N}_{loss}^{dstream} + \mathcal{N}_D}{\mathcal{N}_R}, & \text{otherwise} \end{cases} \quad (3)$$

3) Inferring the SLR: SLR is designed to reflect downstream congestion on a per-RTT basis (*i.e.,* almost realtime downstream congestion), and is computed across all flows from the mbox to the victim. Like LLR, SLR is learned through capabilities returned by the CHM. Specifically, the mbox maintains a hash table (*cTable*) to record capabilities used to learn its SLR. The hash key is the capability itself and the value is a single bit value (initialized to zero) to indicate whether the corresponding key (capability) has been returned.

As described in § V-C2, when a packet arrives (from any source), the mbox stamps a capability on the packet. The capability will be added into *cTable* if it is not a common capability and *cTable*'s length has not reached the predefined threshold $\mathcal{S}_{slr}$. The mbox maintains a timestamp $\mathcal{T}_{slr}$ when the last capability is added into *cTable*. Then, it uses the entire batch of capabilities in the *cTable* to learn the SLR. We set $\mathcal{S}_{slr} = 100$ to allow fast capability loading to the *cTable* (and therefore fast downstream congestion detection), while minimizing sampling error from $\mathcal{S}_{slr}$ being too small.

The mbox allows at most Th_{rtt} from $\mathcal{T}_{slr}$ to receive feedback for all capabilities in the *cTable*. Some capabilities may be returned before $\mathcal{T}_{slr}$ (*i.e.,* before the *cTable* fills). Once a capability in *cTable* is returned, the mbox marks it as received. Upon receiving a new packet with arrival time $t_a > \mathcal{T}_{slr} + Th_{rtt}$, the mbox computes $SLR = \frac{Z_0}{\mathcal{S}_{slr}}$, where Z_0 is the number of *cTable* entries that are not received. The mbox then resets the current *cTable* to be empty to start a new monitoring cycle for learning SLR.

4) Traffic Policing Algorithm: We now detail the traffic policing algorithm to enforce victim-selected traffic control policies. We formalize the traffic policing logic in Algorithm 1. Upon receiving a packet P, the mbox retrieves the entry $\mathcal{F}$ in *iTable* matching P (line 8). If no entry matches, the mbox initializes an entry for P.

P is categorized as a privileged or best-effort packet based on $\mathcal{F}$'s $\mathcal{W}_R$ (line 10). All privileged packets are accepted, whereas best-effort packets are accepted conditionally. If P is privileged, the mbox performs necessary capability handling (line 11) before appending P to the privileged queue. The mbox maintains two FIFO queues to serve all *accepted* packets: the privileged queue serving privileged packets and the best-effort queue serving best-effort packets. The privileged queue has strictly higher priority than the best-effort queue at the output port. CapabilityHandling (line 16) executes the capability generation and *cTable* updates (line 37), as detailed in § V-C2 and § V-C3, respectively.

If P is a best-effort packet, its forwarding decision is subject to the mbox's SLR and $\mathcal{F}'s$ LLR (line 24). If the SLR exceeds Th_{slr}^{drop}, indicating downstream congestion, the mbox discards P. Further, if $\mathcal{F}$'s LLR is already above Th_{slr}^{drop}, the mbox will not deliver best-effort traffic for $\mathcal{F}$ as well since $\mathcal{F}$ already experiences severe losses. Th_{slr}^{drop} is set to be few times larger than a TCP flow's loss rate in normal network condition [32] to absorb burst losses. If the mbox decides to accept P, it performs capability handling (line 27).

Algorithm 1 Traffic Policing Algorithm

```
1  Input:
2    Packet P arrived at time ts;
3  Output:
4    iTable updates and possible cTable updates;
5    The forwarding decision of P;

6  Main Procedure:
7  begin
8    F ← iTableEntryRetrieval(P);
9    F.N_R ← F.N_R + 1;
10   if F.N_R < F.W_R then
       /* Privileged packets                          */
11     CapabilityHandling(P, F);
12     Append P to the privileged queue;
13   else
       /* Best-effort packets                         */
14     BestEffortHandling(P, F);
     /* Starting a new detection period if
        necessary                                      */
15   if ts − F.T_A > D_p then iTableHandling(F);

16 Function: CapabilityHandling(P, F):
17 begin
     /* Two constraints for
        distinct-capability generation                 */
18   if F.P_id < Th_cap and ts−F.T_A < D_p−Th_rtt then
19     F.P_id ← F.P_id + 1;
20     Generate capability C based on Equation (1);
21     cTableHandling(C);
22   else
       /* Common capability for packet
          authentication                               */
23     Generate capability C_c based on Equation (2);

24 Function: BestEffortHandling(P, F):
25 begin
26   if SLR < Th_slr^drop and F.L_R < Th_slr^drop then
27     CapabilityHandling(P, F);
28     Append P to the best-effort queue;
29   else
30     Drop P;    F.N_D ← F.N_D + 1;

31 Function: iTableHandling(F):
32 begin
33   Compute recentLoss based on Equation (3);
     /* Consider the historical loss rate           */
34   F.L_R ← (1 − β) · recentLoss + β · F.L_R;
35   W_R ← BandwidthAllocationPolicy(F);
36   Reset W_V, P_id, N_R and N_D to zero;

37 Function: cTableHandling(C):
38 begin
     /* One batch of cTable is not ready            */
39   if cTable.length < S_slr then
40     Add C into cTable ;
41   if cTable.length == S_slr then  T_slr ← ts ;
```

Finally, if P's arrival triggers a new detection period for $\mathcal{F}$ (line 15), the mbox performs corresponding updates for $\mathcal{F}$ (line 31). To determine $\mathcal{F}$'s LLR, the mbox incorporates both the recent LLR (*recentLoss*) obtained in the current detection period based on Equation (3) and $\mathcal{F}$'s historical loss rate $\mathcal{L}_R$. This design prevents attackers from hiding their previous packet losses via on-off attacks (*e.g.,* shrew attacks [31]). $\mathcal{F}$'s $\mathcal{W}_R$ is updated based on the victim-selected traffic control policies (line 35), as described below.

5) Traffic Control Policies: We list the following representative bandwidth allocation algorithms that may be implemented in the BandwidthAllocationPolicy module to enforce victim-chosen traffic control policies. We clarify that this list is neither complete nor the "best". Instead, these traffic control policies are the typical ones used by prior academic papers. Coming up with industry-driven traffic control policies is not the focus of this paper. Rather, our group is conducting active research in this area.

NaturalShare: for each sender, the mbox sets its $\mathcal{W}_R$ for the next period to the number of delivered packets from the sender in the current period. The design rationale for this policy is that the mbox allows a rate that the sender can sustainably transmit without experiencing a large LLR.

PerSenderFairshare allows the victim to enforce per-sender fair share at bottlenecks. Each mbox fairly allocates its estimated total downstream bandwidth to the senders that reach the victim through the mbox. To this end, the mbox maintains the total downstream bandwidth estimate $\mathcal{N}_{size}^{total}$, which it allocates equally among all senders. To ensure global fairness among all senders, two mboxes sharing the same bottleneck (*i.e.,* the two paths connecting the two mboxes with the victim both traverse the bottleneck link) share their local observations. We design a co-bottleneck detection mechanism using SLR correlation: if two mboxes' observed SLRs are correlated, they share a bottleneck with high probability. In § IX-C, we evaluate the effectiveness of this mechanism.

PerASFairshare is similar to PerSenderFairshare except that the mbox fairly allocates $\mathcal{N}_{size}^{total}$ on a per-AS basis. This policy mimics SIBRA [26], preventing bot-infested ASes from taking bandwidth away from legitimate ASes.

PerASPerSenderFairshare is a hierarchical regime: the mbox first allocates $\mathcal{N}_{size}^{total}$ on a per-AS basis, and then fairly assigns the share obtained by each AS among the AS' senders.

PremiumClientSupport enables the victim to provide premium service to its premium clients, such as bandwidth reservation for upgraded ASes. The victim pre-identifies its premium clients to MiddlePolice. PremiumClientSupport can be implemented together with the aforementioned algorithms.

VI. Packet Filtering

When the victim's IP addresses are kept secret, attackers cannot bypass MiddlePolice's upstream mboxes to route attack traffic directly to the victim. In this case, the downstream packet filtering is *unnecessary* since MiddlePolice can throttle attacks at the upstream mboxes. However, in case of IP address exposure [28], [29], the victim needs to deploy a packet filter to discard bypassing traffic. MiddlePolice designs a filtering mechanism that extends to commodity routers the port-based filtering of previous work [22], [33]. Unlike prior work, the filtering can be deployed upstream of the victim as a commercial service.

A. Traffic Filtering Primitives

Although the MAC-incorporated capability can prove that a packet indeed traverses an mbox, it requires upgrades from deployed commodity routers to perform MAC computation to filter bypassing packets. Thus, we invent a mechanism based on the existing ACL configurations on commodity routers. Specifically, each mbox encapsulates its traversing packets into UDP packets (similar techniques have been applied in VXLAN and [34]), and uses the UDP source and destination ports (a total of 32 bits) to carry an authenticator, which is a shared secret between the mbox and the filtering point. As a result, a 500Gbps attack (the largest attack viewed by Arbor Networks [2]) that uses random port numbers will be reduced to $\sim$100bps since the chance of a correct guess is 2^{-32}. The shared secret can be negotiated periodically based on a cryptographically secure pseudo-random number generator. We do not rely on UDP source address for filtering to since attackers may spoof the mbox's IP address.

B. Packet Filtering Points

Deployed filtering points should have sufficient bandwidth so that the bypassing attack traffic cannot cause packet losses prior to the filtering. The filtering mechanism should be deployed at, or upstream of, each bottleneck link caused by the DDoS attacks. For instance, for a victim with high-bandwidth connectivity, if the bottleneck link is an internal link inside the victim's network, the victim can deploy the filter at the inbound points of its network. If the bottleneck link is the link connecting the victim with its ISP, the victim can can work with its ISP, on commercially reasonable terms, to deploy the filter deeper in the ISP's network such that the bypassing traffic cannot reach the victim's network. Working with the ISPs does not violate the deployment model in § III as MiddlePolice never requires deployment at unrelated ASes. In fact, several ISPs interviewed by us have expressed the interests of offering security services for a new business model.

VII. Source Authentication

MiddlePolice punishes senders even after an offending flow ends. Such persistence can be built on source authentication or any mechanism that maintains sender accountability across flows. As a proof-of-concept, this section sketches a client-transparent source validation mechanism that is specific to HTTP/HTTPS traffic. This mechanism ensures that a sender is on-path to its claimed source IP address.

Our key insight is that the HTTP Host header is in the first few packets of each connection. As a result, the mbox monitors a TCP connection and reads the Host header. If the Host header reflects a generic (not sender-specific) hostname (*e.g.,* victim.com), the mbox intercepts this flow, and redirects the connection (HTTP 302) to a Host containing a token

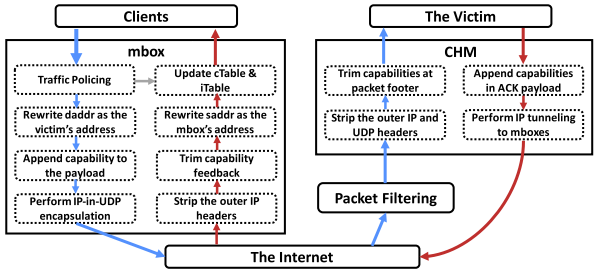

Fig. 2. The software stack of the mbox and CHM.

cryptographically generated from the sender's claimed source address, *e.g.,* T.victim.com, where T is the token. If the sender is on-path, it will receive the redirection, and its further connection will use the sender-specific hostname in the Host header. When an mbox receives a request with a sender-specific Host, it verifies that the Host is proper for the claimed IP source address (if not, the mbox initiates a new redirection), and forwards the request to the victim. Thus, by performing source verification entirely at the mbox, packets from spoofed sources cannot consume any downstream bandwidth from the mbox to the victim.

If the cloud provider hosting the mbox is trusted (for instance, large CDNs have CAs trusted by all major browsers), the victim can share its key with the provider such that HTTPS traffic can be handled in the same way as HTTP traffic. For untrusted cloud providers [22], [35], the mbox should relay the encrypted connection to the victim, which performs Host-header-related operations. The victim terminates unverified senders without returning capabilities to the mbox, so that additional traffic from the unverified senders is best-effort under Algorithm 1. In this case, without relying on the trustworthiness of cloud providers, packets from spoofed sources consume limited downstream bandwidth.

We acknowledge that the above mechanism has its weakness. First, it is only applicable for HTTP/HTTPs traffic, although MiddlePolice itself is not limited to any specific layer-7 application. Second, if the source validation has to be performed by the victim for HTTPS traffic, it is subject to the DoC attack [36] in which attackers flood the victim with numerous new connections to slow down the connection-setup for legitimate clients. Inventing a lightweight anti-spoofing source validation mechanism at the network layer is part of our future work.

VIII. Implementation

We present the implementation of MiddlePolice's prototype in this section. The source code is available at Github [37].

A. The Implementation of mboxes and CHM

The mboxes and the CHM at the victim are implemented based on the NetFilter Linux Kernel Module, which combined have ∼1500 lines of `C` code (excluding the capability generation code). The software stack of our implementation is illustrated in Figure 2.

All inbound traffic from clients to an mbox is subject to the traffic policing whereas only accepted packets go through the capability-related processing. Packet dropping due to traffic policing triggers *iTable* updates. For each accepted packet, the mbox rewrites its destination address as the victim's address to point the packet to the victim. To carry capabilities, rather than defining a new packet header, the mbox appends the capabilities to the end of the original data payload, which avoids compatibility problems at intermediate routers and switches. The CHM is responsible for trimming these capabilities to deliver the original payload to the victim's applications. If the packet filter is deployed, the mbox performs the IP-in-UDP encapsulation, and uses the UDP source and destination port number to carry an authenticator. All checksums need to be recomputed after packet manipulation to ensure correctness. ECN and encapsulation interactions are addressed in [38].

To avoid packet fragmentation due to the additional 68 bytes added by the mbox (20 bytes for outer IP header, 8 bytes for the outer UDP header, and 40 bytes reserved for a capability), the mbox needs to be *a priori* aware of the MTU M_d on its path to the victim. Then the mbox sets its MSS to no more than $M_d-68-40$ (the MSS is 40 less than the MTU). We do not directly set the MTU of the mbox's NIC to rely on the path MTU discovery to find the right packet size because some ISPs may block ICMP packets. On our testbed, $M_d = 1500$, so we set the mbox's MSS to 1360.

Upon receiving packets from upstream mboxes, the CHM strips their outer IP/UDP headers and trims the capabilities. To return these capabilities, the CHM piggybacks capabilities to the payload of ACK packets. To ensure that a capability is returned to the mbox issuing the capability even if the Internet path is asymmetric, the CHM performs IP-in-IP encapsulation to tunnel the ACK packets to the right mbox. We allow one ACK packet to carry multiple capabilities since the victim may generate cumulative ACKs rather than per-packet ACKs. Further, the CHM tries to pack more capabilities in one ACK packet to reduce the capability feedback latency at the CHM. The number of capabilities carried in one ACK packet is stored in the TCP option (the 4-bit `res1` option). Thus, the CHM can append up to 15 capabilities in one ACK packet if the packet has enough space and the CHM has buffered sufficient number of capabilities.

Upon receiving an ACK packet from the CHM, the mbox strips the outer IP header and trims the capability feedback (if any) at the packet footer. Further, the mbox needs to rewrite the ACK packet's source address back to its own address, since the client's TCP connection is expecting to communicate with the mbox. Based on the received capability feedback, the mbox updates the *iTable* and *cTable* accordingly to support the traffic policing algorithm.

B. Capability Generation

We use the AES-128 based CBC-MAC, based on the Intel AES-NI library, to compute MACs, due to its fast speed and availability in modern CPUs [39]. We port the capability implementation (∼400 lines of `C` code) into the mbox and CHM kernel module. The mbox needs to perform both capability generation and verification whereas the CHM performs only verification.

TABLE III

REROUTING TRAFFIC TO Mboxes CAUSES SMALL AS-HOP INFLATION, AND ∼10% ACCESS ASES CAN EVEN REACH THE VICTIM WITH FEWER HOPS THROUGH Mboxes

Victims	N_{infla}^{hop}	P_{cut}^{short}	P_{infla}^{no}
Non-stub ASes	1.1	10.6%	22.2%
Stub ASes	1.5	8.4%	18.0%
Overall	1.3	9.5%	20.1%

IX. EVALUATION

A. The Internet Experiments

This section studies the path length and latency inflation for rerouting clients' traffic to mboxes hosted in the cloud.

1) Path Inflation: We construct the AS level Internet topology based on the CAIDA AS relationships dataset [40], including 52680 ASes and their business relationships [41]. To construct the communication route, two constraints are applied based on the routes export policies in [42] and [43]. First, an AS prefers customer links over peer links and peer links over provider links. Second, a path is valid only if each AS providing transit is paid. Among all valid paths, an AS prefers the path with least AS hops (a random tie breaker is applied if necessary). As an example, we use Amazon EC2 as the cloud provider to host the mboxes, and obtain its AS number based on the report [44]. Amazon claims 11 ASes in the report. We first exclude the ASes not appearing in the global routing table, and find that AS 16509 is the provider for the remaining Amazon ASes, so we use AS 16509 to represent Amazon.

We randomly pick 2000 ASes as victims, and for each victim we randomly pick 1500 access ASes. Among all victims, 1000 victims are stub ASes without direct customers and the remaining victims are non-stub ASes. For each AS-victim pair, we obtain the direct route from the access AS to the victim, and the rerouted path through an mbox. Table III summarizes the route comparison. N_{infla}^{hop} is the average AS-hop inflation of the rerouted path compared with the direct route. P_{cut}^{short} is the percentage of access ASes that can reach the victim with fewer hops after rerouting and P_{infla}^{no} is percentage of ASes without hop inflation.

Overall, it takes an access AS 1.3 more AS-hops to reach the victim after rerouting. Even for stub victims, which are closer the Internet edge, the average hop inflation is only 1.5. We also notice that ∼10% ASes even have *shorter* paths due to the rerouting since a large cloud provider typically have many PoPs that are peered or connected with the Internet backbone.

Besides EC2, we also perform path inflation analysis when mboxes are hosted by CloudFlare. The results show that the average path inflation is about 2.3 AS-hops. For any cloud provider, MiddlePolice has the *same path inflation* as the cloud-based DDoS solutions hosted by the same cloud provider, since capability feedback is carried in ACK packets. As such, deploying MiddlePolice into existing cloud-based systems does not increase path inflation.

2) Latency Inflation: In this section, we study the latency inflation caused by the rerouting. In our prototype running on the Internet, we deploy 3 mboxes on Amazon EC2 (located in North America, Asia and Europe), one victim server in a US university and about one hundred senders (located in North America, Asia and Europe) on PlanetLab [45] nodes. We also deploy few clients on personal computers to test MiddlePolice in home network. The wide distribution of clients allows us to evaluate MiddlePolice on various Internet links. We did not launch DDoS attacks over the Internet, which raises ethical and legal concerns. Instead, we evaluate how MiddlePolice may affect the clients in the normal Internet without attacks, and perform the experiments involving large scale DDoS attacks on our private testbed and in simulation.

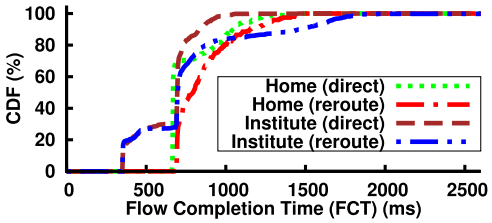

Fig. 3. [Internet] FCTs for direct paths and rerouted paths under various Internet conditions.

In the experiment, each client posts a 100KB file to the server, and its traffic is rerouted to the nearest mbox before reaching the server. We repeat the posting on each client 10,000 times to reduce sampling error. We also run the experiment during both peak hours and midnight (based on the server's timezone) to test various network conditions. As a control, clients post the files to server via direct paths without traversing though mboxes.

Figure 3 shows the CDF of the flow completion times (FCTs) for the file posting. Overall, we notice ∼9% average FCT inflation, and less than 5% latency inflation in home network. Therefore, traffic rerouting introduces small extra latency to the clients.

MiddlePolice's latency inflation includes both rerouting-induced networking latency and capability-induced computational overhead. As evaluated in § IX-B1, the per-packet processing latency overhead caused by capability computation is ∼1.4 μs, which is negligible compared with typical Internet RTTs. Thus, MiddlePolice has latency almost identical to the existing cloud-based DDoS mitigation service.

B. Testbed Experiments

1) Traffic Policing Overhead: In this section, we evaluate the traffic policing overhead on our testbed. We organize three servers as one sender, one mbox and one receiver. All servers, shipped with a quad-core Intel 2.8GHz CPU, run the 3.13.0 Linux kernel. The mbox is installed with multiple Gigabit NICs to connect both the sender and receiver. A long TCP flow is established between the sender and receiver, via the mbox, to measure the throughput achieved by the sender and receiver. To emulate a large number of sources, the mbox creates an *iTable* with N entries. Each packet from the sender triggers a table look up for a random entry. We implement a two-level hash table in the kernel space to reduce the look up latency. Then the mbox generates a capability based on the obtained entry from the table.

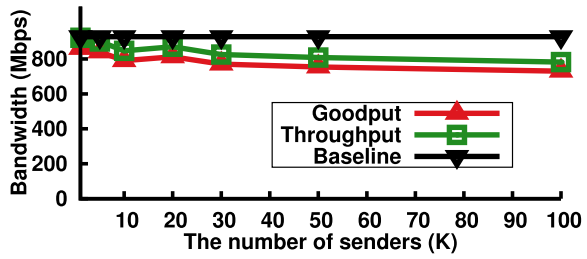

Fig. 4. [Testbed] Throughput and goodput when MiddlePolice polices different numbers of senders.

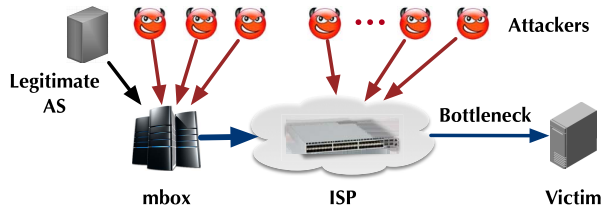

Fig. 5. Testbed network topology.

Figure 4 shows the measured throughput and goodput under various N. The goodput is computed by subtracting the additional header and capability size from the total packet size. The baseline throughput is obtained without MiddlePolice. Overall, the policing overhead in high speed network is small. When a single mbox deals with 100,000 sources sending *simultaneously*, throughput drops by $\sim$10%. Equivalently, MiddlePolice adds around 1.4 microseconds latency to each packet processed. By replicating more geographically distributed mboxes in the cloud, the victim can distribute the workload across many mboxes when facing large scale attacks.

2) Enforce Destination-Defined Traffic Control Policies: We now evaluate MiddlePolice's performance for enforcing victim-defined traffic control policies, along with the effectiveness of filtering bypassing traffic. This section evaluates *pure* MiddlePolice. In reality, once incorporated by existing cloud-based DDoS prevention vendors, MiddlePolice needs only to process traffic that passes their pre-deployed defense.

Testbed topology: Figure 5 illustrates the network topology, including a single-homed victim AS purchasing 1Gbps bandwidth from its ISP, an mbox and 10 access ASes. The ISP is emulated by a Pronto-3297 48-port Gigabit switch to support packet filtering. The mbox is deployed on a server with multiple Gigabit NICs, and each access AS is deployed on a server with a single NIC. We add 100ms latency at the victim via Linux traffic control to emulate the typical Internet RTT. To emulate large scale attacks, 9 ASes are compromised. Attackers adopt a hybrid attack profile: 6 attack ASes directly send large volumes of traffic to the victim, emulating amplification-based attacks, and the remaining attack ASes route traffic through the mbox. Thus, the total volume of attack traffic is 9 times as much as the victim's bottleneck link capacity. Both the inbound and outbound points of the mbox are provisioned with 4Gbps bandwidth to ensure the mbox is not the bottleneck, emulating that the mbox is hosted by the well-provisioned cloud.

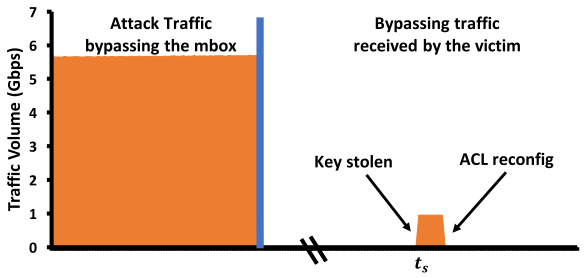

Fig. 6. [Testbed] Packet filtering via ACL.

Packet filtering: We first show the effectiveness of the packet filter. Six attack ASes spoof the mbox's source address and send 6Gbps UDP traffic to the victim. The attack ASes scan all possible UDP port numbers to guess the shared secret. Figure 6 shows the volume of attack traffic bypassing the mbox and its volume received by the victim. As the chance of a correct guess is very small, the filter can effectively stop the bypassing traffic from reaching the victim. Further, even if the shared secret were stolen by attackers at time t_s, the CHM would suddenly receive large numbers of packets without valid capabilities. Since packets traversing the mbox carry capabilities, the CHM realizes that the upstream filtering has been compromised. The victim then re-configures the ACL using a new secret to recover from key compromise. The ACL is effective within few milliseconds after reconfiguration. Thus, the packet filtering mechanism can promptly react to a compromised secret.

NaturalShare and PerASFairshare policies: In this section, we first show that MiddlePolice can enforce both NaturalShare and PerASFairshare policies described in § V-C5. We use the default parameter setting in Table II, and defer detailed parameter study in § IX-C. Since MiddlePolice conditionally allows an AS to send faster than its $\mathcal{W}_R$, we use the *window size*, defined as the larger value between an AS's $\mathcal{W}_R$ and its delivered packets to the victim, as the performance metric. For clear presentation, we normalize the window size to the maximum number of 1.5KB packets deliverable through a 1Gbps link in one detection period. We do not translate window sizes to throughput because packet sizes vary.

Attackers adopt two representative strategies: (i) they send flat rates regardless of packet losses, and (ii) they dynamically adjust their rates based on packet losses (reactive attacks). To launch flat-rate attacks, the attackers keep sending UDP traffic to the victim. The CHM uses a dedicated flow to return received capabilities to the mbox since no ACK packets are generated for UDP traffic. One way of launching reactive attacks is that the attackers simultaneously maintain many more TCP flows than the legitimate AS. Such a many-to-one communication pattern allows the attackers to occupy almost the entire bottleneck, even through each single flow seems completely "legitimate".

The legitimate AS always communicates with the victim via a long-lived TCP connection.

Figure 7 shows the results for the NaturalShare policy. As the bottleneck is flooded by attack traffic, the legitimate AS is forced to enter timeout at the beginning, as illustrated in Figure 7(a). The attackers' window sizes are decreasing over

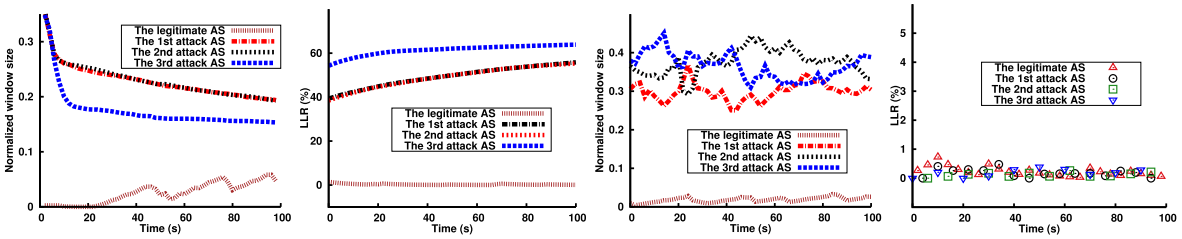

Fig. 7. [Testbed] Enforcing the NaturalShare policy. The legitimate AS gradually obtains a certain amount of bandwidth under flat-rate attacks since attackers' window sizes drop consistently over time (Figure 7(a)) due to their high LLRs (Figure 7(b)). However, the attack ASes can consume over 95% of the bottleneck bandwidth via reactive attacks (Figure 7(c)) while maintaining low LLRs similar to the legitimate AS's LLR (Figure 7(d)).

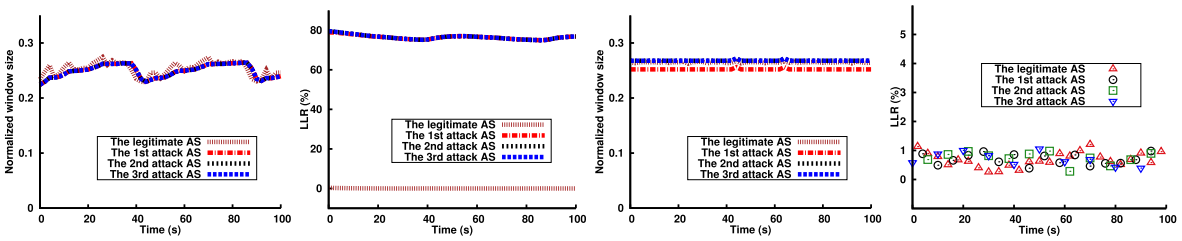

Fig. 8. [Testbed] Enforcing the PerASFairshare policy. The legitimate AS can obtain at least the per-AS fair rate at the bottleneck regardless of the attack strategies (Figures 8(a) and 8(c)). Further, the legitimate AS gains slightly more bandwidth than the attackers under flat-rate attacks as the attack ASes have large LLRs (Figure 8(b)). (a) Window sizes in flat-attacks. (b) LLRs in flat-attacks. (c) Window sizes in reactive att. (d) LLRs in reactive attacks.

time, which can be explained via Figure 7(b). As the volume of attack traffic is well above the bottleneck's capacity, all attack ASes' LLRs are well above Th_{slr}^{drop}. Thus, the mbox drops all their best-effort packets. As a result, when one attack AS's window size is $W(t)$ in detection period t, then $W(t+1) \leq W(t)$ since in period $t+1$ any packet sent beyond $W(t)$ is dropped. Further, any new packet losses from the attack AS, caused by an overflow at the bottleneck buffer, will further reduce $W(t+1)$. Therefore, all attack ASes' window sizes are consistently decreasing over time, creating spare bandwidth at the bottleneck for the legitimate AS. As showed in Figure 7(a), the legitimate AS gradually recovers from timeouts.

The NaturalShare policy, however, cannot well protect the legitimate AS if the attackers adopt the reactive attack strategy. By adjusting the sending rates based on packet losses, the attack ASes can keep their LLRs low enough to regain the advantage of delivering best-effort packets. Meanwhile, they can gain much more bandwidth by initiating more TCP flows. Figure 7(c) shows the window sizes when each attack AS starts 200 TCP flows whereas the legitimate AS has only one. The attackers consume over 95% of the bottleneck bandwidth, while keeping low LLRs similar to that of the legitimate AS (as shown in Figure 7(d)).

Figure 8 shows the results for the PerASFairshare policy. Figures 8(c) and 8(c) demonstrate that the legitimate AS receives at least per-AS fair rate at the bottleneck regardless of the attack strategies, overcoming the shortcomings of the NaturalShare policy. Further, under flat-rate attacks, the legitimate AS has slightly larger window sizes than the attackers since, again, the mbox does not accept any best-effort packets from the attackers due to their high LLRs (Figure 8(b)).

PremiumClientSupport policy: This section evaluates the PremiumClientSupport policy discussed in § V-C5.

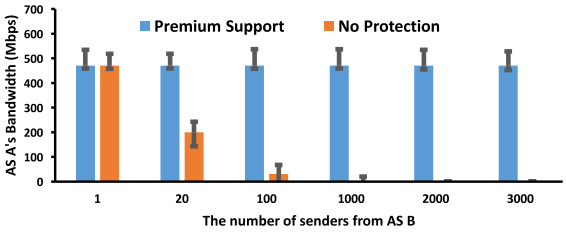

Fig. 9. [Testbed] MiddlePolice ensures that the premium client (AS A) receives consistent bandwidth.

We consider a legitimate AS (AS A) that is a premium client which reserves half of the bottleneck bandwidth. Figure 9 plots AS A's bandwidth when the number of senders from the attack ASes increases. With the PremiumClientSupport policy, MiddlePolice ensures AS A receives consistent bandwidth regardless of the number of senders from the attack ASes. However, without such a policy, the attack ASes can selfishly take away the majority of bottleneck bandwidth by involving more senders. In reality, since an adversary can control large number of bots, the PremiumClientSupport policy turns out to be invaluable according to our interviews with industry people.

C. Large Scale Evaluation

In this section, we further evaluate MiddlePolice via large scale simulations on ns-3 [46]. We desire to emulate real-world DDoS attacks in which up to millions of bots flood a victim. To circumvent the scalability problem of ns-3 at such a scale, we adopt the same approach in NetFence [13], *i.e.,* by fixing the number of nodes (∼5000) and scaling down the link capacity proportionally, we can simulate attack scenarios where 1 million to 10 million attackers flood a 40 Gbps link. The simulation topology is similar to the testbed topology, except that all attackers are connected to the mbox.

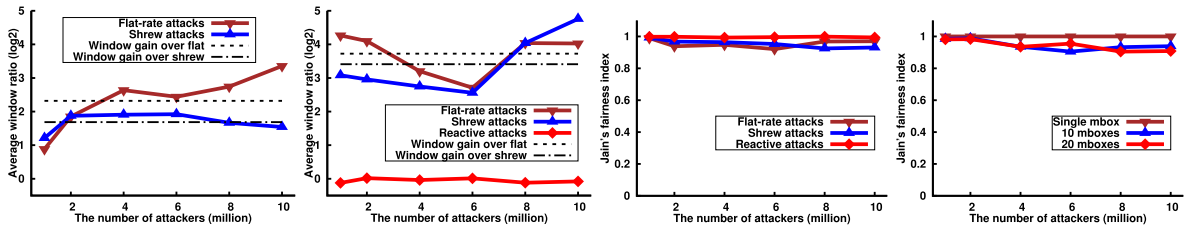

Fig. 10. [Simulation] Evaluating NaturalShare & PerSenderFairshare in large scale. Figures 10(a) and 10(b) show that the clients' average window size is larger than that of the attackers under both flat-rate and shrew attacks. Figure 10(c) proves that the clients' window sizes converge to fairness in the PerSenderFairshare policy. Figure 10(d) shows that MiddlePolice can enforce strong fairness among all senders even without coordination among the mboxes.

Besides the flat-rate attacks and reactive attacks, we also consider the on-off shrew attacks [31] in the simulations. Both the on-period and off-period in shrew attacks are 1s. The number of attackers is 10 times larger than that of legitimate clients. In flat-rate attacks and shrew attacks, the attack traffic volume is 3 times larger than the capacity of the bottleneck. In reactive attacks, each attacker opens 10 connections, whereas a client has one. The bottleneck router buffer size is determined based on [47], and the RTT is 100ms.

NaturalShare & PerSenderFairshare in Scale: Figure 10 shows the results for enforcing NaturalShare and PerSenderFairshare policies with default parameter settings. We plot the ratio of clients' average window size to attackers' average window size for the NaturalShare policy in Figure 10(a). For flat-rate attacks and shrew attacks, it may be surprising that the clients' average window size is larger than that of the attackers. Detailed trace analysis shows that it is because that the window sizes of a large portion of attackers keep decreasing, as we explained in our testbed experiment. As the number of attackers is much larger than the client count, the attackers' average window size turns out to be smaller than that of the clients, although the absolute volume of attack traffic may be still higher. Under reactive attacks, the clients' average window size is almost zero, which is outside the plot scope of Figure 10(a).

Figure 10(b) shows that the clients enjoy even larger window ratio gains under the PerSenderFairshare policy in flat-rate and shrew attacks because even more attackers enter the window dropping mode. Further, the PerSenderFairshare ensures that the clients' average window size is close to the per-sender fair rate in reactive attacks. Figure 10(c) demonstrates such per-client fairness since Jain's fairness index [48] (FI) is close to 1.

mbox Coordination for Co-Bottleneck Detection: To enforce global per-sender fairness, the mboxes sharing the same bottleneck link share their local observations (§V-C5). We first investigate how bad the FI can be without such inter-mbox coordination. We reconstruct the topology to create multiple mboxes, and map each client to a random mbox. The attackers launch reactive attacks. The results, plotted in Figure 10(d), show that the FI drops slightly, by ∼8%, even if 20 mboxes make local rate allocations without performing any coordination among them.

To complete our design, we further propose the following co-bottleneck detection mechanism. The design rationale is that if two mboxes' SLR observations are correlated, they share a bottleneck with high probability. To validate this, we rebuild the network topology to create the scenarios where two mboxes share and do not share a bottleneck, and study the correlation coefficient of their SLRs. We compute one coefficient for every 100 SLR measurements from each mbox. Figure 11 shows the CDF of the coefficient. Clearly, the coefficient reflects whether the two mboxes share a bottleneck. Thus, by continuously observing such correlation between two mboxes' SLRs, MiddlePolice can determine with increasing certainty whether or not they share a bottleneck, and can configure their coordination accordingly.

TABLE IV

[SIMULATION] CLIENTS' AVERAGE WINDOW SIZE UNDER DIFFERENT PARAMETER SETTINGS. (a) THE NaturalShare POLICY. (b) THE PerSenderFairshare POLICY

(a)

	$\mathcal{D}_p$		Th_{slr}^{drop}		β	
	2s	8s	0.03	0.1	0.5	0.9
Flat	1.1	0.17	0.78	0.39	1.1	0.78
Shrew	1.3	0.65	0.77	1.0	1.2	0.80

(b)

	$\mathcal{D}_p$		Th_{slr}^{drop}		β	
	2s	8s	0.03	0.1	0.5	0.9
Flat	1.0	1.1	1.0	0.69	0.85	0.81
Shrew	1.1	0.98	0.72	0.83	1.0	0.98
Reactive	1.0	0.99	1.0	0.94	1.0	1.0

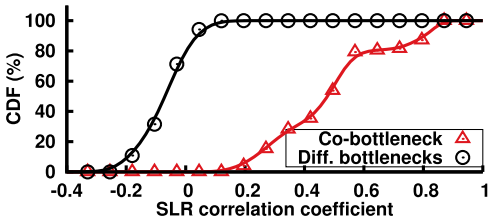

Fig. 11. [Simulation] The SLR correlation coefficient reflects whether two mboxes share a bottleneck link.

Parameter Study: In this section, we evaluate MiddlePolice using different parameters than the default values in Table II. We mainly focus on $\mathcal{D}_p$, Th_{slr}^{drop} and β. For each parameter, we vary its value in experiments to obtain the clients' average window size under the 10-million bot attack. The results showed in Table IV are normalized to the window sizes obtained using default parameters in Table II.

Under the NaturalShare policy, the shorter $\mathcal{D}_p$ produces a larger window size for legitimate clients since each sender's

TABLE V

PROPERTY COMPARISON WITH OTHER RESEARCH PROPOSALS. "$O(N)$ STATES" MEANS THAT THE NUMBER OF STATES MAINTAINED BY A ROUTER INCREASES WITH THE NUMBER OF ATTACKERS. "CRYPTOGRAPHY" MEANS THAT A ROUTER NEEDS TO SUPPORT CRYPTOGRAPHY OPERATION, *e.g.,* MAC COMPUTATION. "PUZZLE" MEANS THAT THE MECHANISM REQUIRES COMPUTATIONAL PUZZLE DISTRIBUTION

	Pushback[8]	SIFF[11], TVA[12]	Netfence[13]	Phalanx[14]	Mirage[21]	SIBRA[26]	MiddlePolice
Source upgrades	**No**	Yes	Yes	Yes	Yes	Yes	**No**
Dest. upgrades	**No**	Yes	Yes	Yes	Yes	Yes	Yes
AS deployment	Unrelated	Unrelated	Unrelated	Unrelated	**Related**	Unrelated	**Related**
Router support	$O(N)$ states	Cryptography; $O(N)$ states for [12]	$O(N)$ states; Cryptography	$O(N)$ states	Larger memory	**None**	**None**
Traffic control policies	None	None	Per-sender fairness	None	Per-compute fairness	Per-AS fairness	**Victim-defined policies**
Other requirements	**None**	New header	New header; Passport[50]	New header	Puzzle; IPv6 upgrade	Redesign the Internet	**None**

$\mathcal{W}_R$ is updated per-period so that a smaller $\mathcal{D}_p$ causes faster cut in attackers' window sizes. For Th_{slr}^{drop}, a smaller value slows down the clients' recovery whereas a larger value allows larger window sizes for attackers. Both will reduce the clients' share. A larger β has negative effects as it takes more time for the clients to recover to a low LLR.

With the PerSenderFairshare policy, MiddlePolice's performance is more consistent under different parameter settings. The most sensitive parameter is Th_{slr}^{drop} because it determines whether one source can send best-effort traffic.

X. RELATED WORK

In this section, we briefly discuss previous academic work. Previous research approaches can be generally categorized into capability-based approaches (SIFF [11], TVA [12], NetFence [13]), filtering-based approaches (Traceback [5], [6], AITF [7], Pushback [8], [9], StopIt [10]), overlay-based approaches (Phalanx [14], SOS [15], Mayday [16]), deployment-friendly approaches (Mirage [21], CRAFT [24]), approaches based on future Internet architectures (SCION [17], SIBRA [26], XIA [19], AIP [18]), and others (SpeakUp [20], SDN-based [50], [51], CDN-based [22]). We summarize the properties of one or two approaches from each category in Table V. The comparison shows that MiddlePolice requires the *least* deployment (no source upgrades, no additional router support and no deployment from unrelated ASes) while providing the *strongest* property (enforcing destination-chosen traffic control policies).

XI. DISCUSSION

mboxes Mapping: MiddlePolice can leverage the end-user mapping [52] to achieve better mbox assignment, such as redirecting clients to the nearest mbox, mapping clients according to their ASes, and load balancing.

Incorporating Endhost Defense: MiddlePolice can cooperate with the DDoS defense mechanism deployed, if any, on the victim. For instance, via botnet identification [53], [54], the victim can instruct the mboxes to block botnet traffic early at upstream so as to save more downstream bandwidth for clients. Such benefits are possible because the traffic control policies enforced by MiddlePolice are completely driven by designation servers.

Additional Monetary Cost: As discussed in § IX-B1, MiddlePolice introduces small computational overhead. Compared with basic DDoS-as-a-service solutions, MiddlePolice offers additional functionalities such as enabling destination-chosen policies and filtering bypassing traffic. In a competitive marketplace, service's price (the monetary cost) should scale with the cost of providing that service, which, in the case of MiddlePolice, is low.

Traffic Learning Tools: Through our interviews with industry people, we notice that some organizations have limited knowledge about their traffic. Thus, although MiddlePolice is able to enforce arbitrary victim-selectable traffic control policies, an organization would receive limited benefits if it could not figure out correct policies (*e.g.,* what typical of traffic is most critical to their business, which clients are more important to their profit). To assist these organizations to better understand their network traffic and therefore bring out more rational policies, we are in active research on inventing multiple machine learning based traffic learning tools, focusing on traffic classification and client clustering.

XII. CONCLUSION

Guided by our large-scale industrial interviews with potential DDoS victims, this paper presents MiddlePolice, the first DDoS mitigation system that offers readily deployable and proactive DDoS prevention. In its design, MiddlePolice explicitly addresses three challenges. First, MiddlePolice designs a capability mechanism that requires only limited deployment from the cloud, rather than widespread Internet upgrades. Second, MiddlePolice is fully destination-driven, addressing the shortcomings of the existing DDoS prevention systems that can only work either protocol or vendor defined traffic control policies. Finally, MiddlePolice addresses the traffic-bypass vulnerability of the existing cloud-based solutions. Extensive evaluations on the Internet, testbed and large scale simulations validate MiddlePolice's deployability and effectiveness in enforcing detestation-chosen traffic control policies. Besides the technical contribution, based our large-scale survey, we also articulated the design space of future advanced DDoS prevention mechanism, though which we hope to offer more insight into practical DDoS mitigation in research community.

ACKNOWLEDGMENTS

The authors thank the anonymous ACM CCS and ACM/IEEE TON reviewers for their valuable feedback.

REFERENCES

[1] 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, pp. 1268–1279.

[2] A. Networks. (2016). *Worldwide Infrastructure Security Report*, vol. 9. [Online]. Available: https://www.arbornetworks.com/images/documents/WISR2016_EN_Web.pdf

[3] B. Krebs. *KrebsOnSecurity Hit With Record DDoS*. Accessed: 2017. [Online]. Available: https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos/

[4] O. Klaba. *OVH Hit With 1 Tbps DDoS*. Accessed: 2017. [Online]. Available: https://twitter.com/olesovhcom/status/778830571677978624

[5] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical network support for IP traceback," in *Proc. ACM SIGCOMM*, 2000, pp. 295–306.

[6] D. X. Song and A. Perrig, "Advanced and authenticated marking schemes for IP traceback," in *Proc. IEEE INFOCOM*, Apr. 2001, pp. 878–886.

[7] K. J. Argyraki and D. R. Cheriton, "Active Internet traffic filtering: Real-time response to denial-of-service attacks," in *Proc. USENIX ATC*, 2005, pp. 135–148.

[8] R. Mahajan *et al.*, "Controlling high bandwidth aggregates in the network," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 3, pp. 62–73, 2002.

[9] J. Ioannidis and S. M. Bellovin, "Implementing pushback: Router-based defense against DDoS attacks," in *Proc. USENIX NSDI*, 2002, pp. 1–8.

[10] 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, pp. 195–206.

[11] A. Yaar, A. Perrig, and D. Song, "SIFF: A stateless Internet flow filter to mitigate DDoS flooding attacks," in *Proc. IEEE SP*, May 2004, pp. 130–143.

[12] X. Yang, D. Wetherall, and T. Anderson, "A DoS-limiting network architecture," in *Proc. ACM SIGCOMM*, 2005, pp. 241–252.

[13] X. Liu, X. Yang, and Y. Xia, "NetFence: Preventing Internet denial of service from inside out," in *Proc. ACM SIGCOMM*, 2011, pp. 255–266.

[14] C. Dixon, T. E. Anderson, and A. Krishnamurthy, "Phalanx: Withstanding multimillion-node botnets," in *Proc. USENIX NSDI*, 2008, pp. 45–58.

[15] A. D. Keromytis, V. Misra, and D. Rubenstein, "SOS: Secure overlay services," in *Proc. ACM SIGCOMM*, 2002, pp. 61–72.

[16] D. G. Andersen, "Mayday: Distributed filtering for Internet services," in *Proc. USNIX USITS*, 2003, pp. 20–30.

[17] X. Zhang, H.-C. Hsiao, G. Hasker, H. Chan, A. Perrig, and D. G. Andersen, "SCION: Scalability, control, and isolation on next-generation networks," in *Proc. IEEE SP*, May 2011, pp. 212–227.

[18] D. G. Andersen *et al.*, "Accountable Internet protocol (AIP)," in *Proc. ACM SIGCOMM*, 2008, pp. 339–350.

[19] D. Naylor *et al.*, "XIA: Architecting a more trustworthy and evolvable Internet," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 50–57, 2014.

[20] M. Walfish *et al.*, "DDoS defense by offense," in *Proc. ACM SIGCOMM*, 2006, pp. 303–314.

[21] P. Mittal, D. Kim, Y.-C. Hu, and M. Caesar. (2011). "Mirage: Towards deployable DDoS defense for Web applications." [Online]. Available: https://arxiv.org/abs/1110.1060

[22] Y. Gilad, A. Herzberg, M. Sudkovitch, and M. Goberman, "CDN-on-demand: An affordable DDoS defense via untrusted clouds," in *Proc. NDSS*, 2016, pp. 1–15.

[23] *AT&T Denial of Service Protection*. Accessed: 2016. [Online]. Available: http://soc.att.com/1IIlUec and http://www.webcitation.org/6jnDvRIgB

[24] D. Kim, J. T. Chiang, Y.-C. Hu, A. Perrig, and P. Kumar, "CRAFT: A new secure congestion control architecture," in *Proc. ACM CCS*, 2010, pp. 705–707.

[25] B. Parno *et al.*, "Portcullis: Protecting connection setup from denial-of-capability attacks," in *Proc. ACM SIGCOMM*, 2007, pp. 289–300.

[26] C. Basescu *et al.*, "SIBRA: Scalable Internet bandwidth reservation architecture," in *Proc. NDSS*, 2016, pp. 1–16.

[27] S. Peter *et al.*, "One tunnel is (often) enough," in *Proc. ACM SIGCOMM*, 2014, pp. 99–110.

[28] T. T. Miu *et al.*, "Universal DDoS mitigation bypass," in *Proc. Black Hat USA*, 2013, pp. 1–15.

[29] T. Vissers, T. Van Goethem, W. Joosen, and N. Nikiforakis, "Maneuvering around clouds: Bypassing cloud-based security providers," in *Proc. ACM CCS*, 2015, pp. 1530–1541.

[30] M. Kührer, T. Hupperich, C. Rossow, and T. Holz, "Exit from hell? Reducing the impact of amplification DDoS attacks," in *Proc. USENIX Secur. Symp.*, 2014, pp. 111–125.

[31] A. Kuzmanovic and E. W. Knightly, "Low-rate TCP-targeted denial of service attacks: The shrew vs. the mice and elephants," in *Proc. ACM SIGCOMM*, 2003, pp. 75–86.

[32] S. Sundaresan *et al.*, "Broadband Internet performance: A view from the gateway," in *Proc. ACM SIGCOMM*, 2011, pp. 134–145.

[33] Y. Gilad and A. Herzberg, "LOT: A defense against IP spoofing and flooding attacks," *ACM Trans. Inf. Syst. Secur.*, vol. 15, no. 2, 2012, Art. no. 6.

[34] T. Herbert, "UDP encapsulation in linux," in *Proc. Tech. Conf. Linux Netw.*, 2015, pp. 1–44.

[35] J. Liang, J. Jiang, H. Duan, K. Li, T. Wan, and J. Wu, "When HTTPS meets CDN: A case of authentication in delegated service," in *Proc. IEEE SP*, May 2014, pp. 67–82.

[36] K. Argyraki and D. Cheriton, "Network capabilities: The good, the bad and the ugly," *ACM HotNets-IV*, 2005, pp. 1–5.

[37] *MiddlePolice Source Code*. (2018). [Online]. Available: https://github.com/zliuInspire/MiddlePolice

[38] B. Briscoe, *Tunnelling of Explicit Congestion Notification*, document RFC 6040, 2010.

[39] S. Gueron, "Intel advanced encryption standard (AES) instructions set," Intel, White Paper, 2010.

[40] *AS Relationships—CIDR Report*. Accessed: Dec. 2015. [Online]. Available: http://www.caida.org/data/as-relationships/ and http://www.webcitation.org/6jnEGOQwj

[41] X. Dimitropoulos *et al.*, "AS relationships: Inference and validation," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 1, pp. 29–40, 2007.

[42] L. Gao, T. G. Griffin, and J. Rexford, "Inherently safe backup routing with BGP," in *Proc. IEEE INFOCOM*, Apr. 2001, pp. 547–556.

[43] S. Goldberg, M. Schapira, P. Hummon, and J. Rexford, "How secure are secure interdomain routing protocols," in *Proc. ACM SIGCOMM*, 2010, pp. 87–98.

[44] *AS Names—CIDR Report*. Accessed: 2015. [Online]. Available: http://www.cidr-report.org/as2.0/autnums.html

[45] *PlanetLab*. Accessed: 2016. [Online]. Available: https://www.planet-lab.org/ and http://www.webcitation.org/6jnE8SIOE

[46] *NS-3: A Discrete-Event Network Simulator*. Accessed: 2016. [Online]. Available: http://www.nsnam.org/

[47] G. Appenzeller, I. Keslassy, and N. McKeown, "Sizing router buffers," in *Proc. ACM SIGCOMM*, 2004, pp. 281–292.

[48] D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Comput. Netw. ISDN Syst.*, vol. 17, no. 1, pp. 1–14, 1989.

[49] X. Liu, A. Li, X. Yang, and D. Wetherall, "Passport: Secure and adoptable source authentication," in *Proc. USENIX NSDI*, 2008, pp. 365–378.

[50] S. K. Fayaz, Y. Tobioka, V. Sekar, and M. Bailey, "Bohatei: Flexible and elastic DDoS defense," in *Proc. USENIX Secur. Symp.*, 2015, pp. 817–832.

[51] S. Shin *et al.*, "FRESCO: Modular composable security services for software-defined networks," in *Proc. NDSS*, 2013, pp. 1–16.

[52] F. Chen, R. K. Sitaraman, and M. Torres, "End-user mapping: Next generation request routing for content delivery," in *Proc. ACM SIGCOMM*, 2015, pp. 167–181.

[53] C. Livadas, R. Walsh, D. Lapsley, and W. T. Strayer, "Usilng machine learning technliques to identify botnet traffic," in *Proc. IEEE LCN*, Nov. 2006, pp. 967–974.

[54] A. Karasaridis, B. Rexroad, and D. Hoeflin, "Wide-scale botnet detection and characterization," in *Proc. USENIX HotBots*, 2007, pp. 1–8.

Zhuotao Liu, photograph and biography not available at the time of publication.

Hao Jin, photograph and biography not available at the time of publication.

Yih-Chun Hu, photograph and biography not available at the time of publication.

Michael Bailey, photograph and biography not available at the time of publication.