Leveraging Prefix Structure to Detect Volumetric DDoS Attack Signatures with Programmable Switches

Chris Misa*, Ramakrishnan Durairajan*, Arpit Gupta†, Reza Rejaie*, and Walter Willinger‡
*University of Oregon †UCSB ‡NIKSUN, Inc.

***Abstract*—As increasingly complex and dynamic volumetric DDoS attacks continue to wreak havoc on edge networks, two recent developments promise to bolster DDoS defense at the edge. First, programmable switches have emerged as promising means for achieving scalable and cost-effective attack signature detection. However, their practical application in edge networks remains a challenging open problem. Second, machine learning (ML)-based solutions have demonstrated potential in accurately detecting attack signatures based on per-flow traffic features. Yet, their inability to effectively scale to the traffic volumes and number of flows in actual production edge networks has largely excluded them from practical considerations.**

In this paper, we introduce ZAPDOS, a novel approach to accurately, quickly, and scalably detect volumetric DDoS attack signatures at the source prefix level. ZAPDOS is the first to utilize a key characteristic of the observed structure of measured attack and benign source prefixes (*i.e.,* a pronounced cluster-within-cluster property) and effectively apply it in practice against modern attacks. ZAPDOS operates by monitoring aggregate prefix-level features in switch hardware, employing a learning model to identify prefixes suspected of containing attack sources, and using several innovative algorithmic methods to pinpoint attack sources efficiently. We have built a hardware prototype of ZAPDOS and a packet-level software simulator which achieve comparable accuracy results. Since existing datasets are inadequate for training and evaluating prefix-level models, we have developed a new data-fusion methodology for training and evaluating ZAPDOS. We use our prototype and simulator to show that ZAPDOS can detect volumetric DDoS attack signatures with *orders of magnitude* lower error rates than state-of-the-art under comparable monitoring resource budgets and for a range of different attack scenarios.

***Index Terms*—Network security and measurement, programmable switch hardware, DDoS defense**

1. Introduction

Distributed Denial of Service (DDoS) attacks continue to be one of the most pervasive threats to online services and service providers alike. Of particular concern in the broader class of DDoS are *volumetric* attacks that can render critical services or entire networks unreachable through massive floods of traffic (*e.g.,* 3.5 Tbps [1]) using attack vectors like DNS amplification/reflection [2]–[5] or botnets [6]–[8]. Such attacks can result in loss of customers and revenue for the victims (*e.g.,* financial institutions, universities, and local governments) and waste valuable network resources due to large volumes of unproductive traffic traversing the Internet.

The first critical step in defending against volumetric DDoS is detecting *attack signatures* to separate attack from benign traffic. For example, an attack signature could be a list of source IP addresses observed in attack traffic along with the particular attack vector(s) observed from each source. Given increasingly massive traffic volumes, systems that detect attack signatures must be highly resource efficient and able to scale to high traffic rates as well as large traffic complexities (*e.g.,* number of distinct sources). Furthermore, in addition to multiple attack vectors, modern DDoS attacks leverage attack dynamics aimed at evading simple signature detection methods (*e.g.,* changing attack vectors and/or sources over time in a single attack), implying signature detection must be a dynamic and flexible process.

Recent efforts develop volumetric DDoS defenses for large ISPs (*e.g.,* AT&T) or CDNs (*e.g.,* Cloudflare) but fail to address signature detection requirements in a variety of edge network scenarios (*e.g.,* small- or medium-sized enterprises) [9]. Such networks face tighter resource constraints (*e.g.,* monitoring for attack signatures on only a few border switches) and rely on upstream third-party mitigation (*e.g.,* upstream cloud- or service-providers [10], [11], IXPs [12], [13], or scrubbers [14], [15]) since their border links may in fact be the DDoS target. On the one hand, efforts that leverage programmable switch hardware [16]–[19] can reduce the resource overheads by orders of magnitude, but fail to produce actionable attack signatures (due to tight binding of signature detection and mitigation actions) and only implement simple heuristic- or threshold-based detection with limited accuracy [20]–[23]. On the other hand, efforts to leverage machine learning (ML) techniques can achieve high accuracy but rely on large vectors of features computed for every traffic source or flow (attack as well as benign), leading to infeasible resource overheads [24]–[27].

Rather than settling for limited accuracy or excessive resource overheads, we propose to leverage a key yet underexploited insight in this work: *observed IP addresses in network traffic exhibit a pronounced cluster-within-cluster behavior when viewed as hierarchical prefix trees,* irrespective of whether the traffic is benign [28], [29] or associated

Published in *IEEE Symposium on Security and Privacy*.

with attacks [30]. Whereas other recent volumetric DDoS signature detection efforts tacitly assume a uniform distribution of attack and benign sources by applying sketch-based approximation or classifying every possible source, we posit instead that attack signatures should be constructed at *prefix-level* to reflect and exploit inherent clustering of addresses. In particular, prefix-level signatures are able to achieve the same accuracy as source-level signatures while requiring a fraction of the monitoring resources. As a result, when working at prefix-level, larger feature vectors and more complex ML-based modeling techniques can be applied over fewer observations, thus reducing overheads.

Building on this insight, we combine efficient switch-hardware-based feature gathering with accurate CPU-based ML modeling in a novel prefix-level volumetric DDoS attack signature detection approach called *ZAPDOS*.[1] In particular, *ZAPDOS* addresses the following challenges.

- Prefix aggregates inherently contain a distinctive blend of attack and benign traffic features, which is not captured in existing datasets and modeling approaches. We develop a novel data-fusion method to generate a large number of attack scenarios with realistic packet-level and prefix-level distributions. We train classification models on these scenarios using a prefix-length weighting method to tune the model for improved performance (§ 4).
- Although switch hardware can only monitor a fixed number of traffic entities (due to limited TCAM and SRAM entries), optimal prefix-level attack signatures require a variable number of prefixes at variable lengths. *ZAPDOS* develops a novel scheduling-based iterative refinement algorithm that works in a windowed fashion, monitoring a fixed number of prefixes in each time window, and dynamically deciding the length of each prefix included in the attack signature to avoid reporting prefixes containing benign traffic (§ 5).
- Iterative refinement approaches must be robust against changes in attack sources and vectors aimed at foiling signature detection. *ZAPDOS* includes novel look-ahead and look-back algorithmic components that respectively increase the refinement speed and update the refinement focus when attack traffic changes (§ 6).

We implement a prototype of ZAPDOS for Tofino-based switches [18], which includes several solutions to reduce latency overheads (§ 7). Using an independent test set of attack scenarios generated with our data-fusion methodology, we demonstrate our prototype can quickly generate accurate attack signatures (*e.g.,* detecting 99% of attack traffic in less than 25 s). We also develop a packet-level simulator of *ZAPDOS* for evaluating against a wider range of complex and dynamic volumetric DDoS attacks. Using this simulator, we show that *ZAPDOS* can accurately detect such complex and dynamic attack traffic at scale. Specifically, compared to state-of-the-art DDoS defenses such as Jaqen [20] or Euclid [21], *ZAPDOS* achieves two to six orders of magnitude better accuracy when attack sources are realistically clustered in the IPv4 space (§ 8).

1. *ZAPDOS* stands for Zooming-in At Prefix-level DdOs Signatures.

To encourage further research in this area, we open-source our packet-level simulator, examples of generated attack traffic traces, and implementation of our data-fusion method at https://onrg.gitlab.io/projects/zapdos/.

2. Background & Motivation

2.1. Setting

Edge networks such as enterprises, campuses, regional ISPs, and other medium-to-small-sized ISPs (*e.g.,* as shown in Figure 1) suffer from volumetric DDoS attacks in three distinct phases.

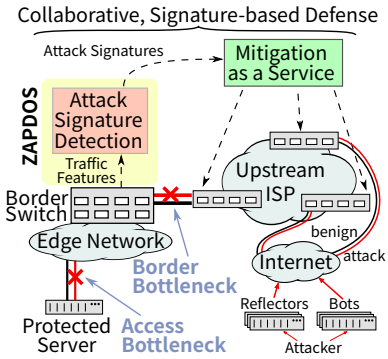


Figure 1: Example edge network showing two possible bottleneck links that could be flooded by a volumetric DDoS.

Pre-attack phase. Based on risk assessment (*e.g.,* planning for a mission-critical event) edge networks typically make preparations for dealing with DDoS attacks such as deploying traffic monitoring systems, securing backup lines of communication, and procuring options to deploy upstream DDoS mitigation (*e.g.,* block lists in upstream ISP).

Active-attack phase. During a volumetric DDoS attack, an adversary sends a large volume of attack traffic which immobilizes the edge network by flooding a bottleneck link as shown in Figure 1. The edge network must quickly generate a signature that identifies attack traffic and coordinate the deployment of mitigation with upstream services to reduce the volume of attack traffic and maintain network connectivity.

Post-attack phase. After the attack event, edge networks must analyze the event, including careful inspection of the attack signature, in order to prepare for future attacks and to understand any other malicious actions for which the attack may have provided cover (*e.g.,* infiltration, data exfiltration).

ZAPDOS primarily focuses on quickly and efficiently detecting volumetric DDoS attack signatures during the active-attack phase, though it relies on traffic monitoring deployed in the pre-attack phase, and the signatures it generates may also be useful in the post-attack phase.

2.2. Requirements

The process of quickly generating a signature that can effectively support mitigation of volumetric DDoS attacks on edge networks implies the following three requirements.

R1: Accurate attack signatures. Attack signatures must accurately separate volumetric DDoS attack traffic from normal benign traffic. They must be expressed in a form intelligible to network administrators and to upstream mitigation services (*e.g.,* specifying attack sources, L3/L4 protocol of attack packets, etc.). In existing proposed switch hardware DDoS defense systems [20], [21] it is hard if not impossible for network administrators to gain visibility into which subsets of traffic are affected by deployed mitigation.
R2: Scalable resource efficiency. Given limited traffic processing resources in edge networks, volumetric DDoS signature detection must remain highly resource efficient as traffic volume and number of attack sources increase. Existing proposals which leverage programmable switch hardware [20], [31] demonstrate orders of magnitude reduction of overheads w.r.t. traffic volume, but can only implement heuristic and approximate detection with limited accuracy. Higher accuracy ML-based proposals [24]–[26] require infeasible traffic monitoring overheads to collect fine-grained features for large numbers of flows. Edge networks must carefully allocate limited resources (*e.g.,* TCAM, SRAM) for signature detection without compromising on signature accuracy or flexibility.
R3: Robust against multiple vectors and dynamic attacks. In addition to a large list of well-established vectors for generating volumetric DDoS attack traffic (*e.g.,* reflection [32]–[35] or botnets [7], [8], [36]), modern adversaries also combine multiple vectors and dynamically change attack traffic during attack episodes. As a result, signature detection must be able to capture a wide variety of potential attack vectors and to quickly update detected signatures as the underlying attack traffic changes.

2.3. Related Work

As shown in Table 1, a number of recent research efforts have addressed some of these requirements using sketch-based switch hardware, flow-level ML models, or prefix-level refinement algorithms, but none have succeeded in simultaneously satisfying all requirements.

Key leverage	Defense	R1 (Accurate)	R2 (Scalable)	R3 (Robust)
Sketch, switch hardware	Jaqen [20]		✓	partial
	Euclid [21]		✓	
Flow-based ML	LUCID [24]	✓		
Pfx. refinement	RADAR [37]		✓	
All of above	***ZAPDOS***	✓	✓	✓

TABLE 1: Overview of how existing defenses compare against the requirements set out in § 2.1.

Programmable switch hardware for resource efficiency. Recent studies [20], [21], [23], [38]–[40] implement combined attack signature detection and mitigation policies directly in switch hardware by using probabilistic data structures known as sketches [41], [42] (thus satisfying **R2**). Some of these efforts (*e.g.,* Jaqen [20] and Poseidon [38]) also partially satisfy **R3** by developing libraries for detecting a range of modern attack vectors. Other efforts rely on behavioral assumptions that are easier for modern attacks to circumvent (*e.g.,* Euclid [21] assumes that attacks can be identified by their contribution towards the entropy of observed addresses).

However, a key limitation of sketch-based efforts is that they cannot report detailed attack signatures because they do not keep track of flow keys, *i.e.,* they fail to satisfy **R1**. Even if one could extract the attack signature from a sketch-based method (*e.g.,* using "reversible" sketch techniques [43]–[45]), the types of metrics and decisions that can be made using sketch-based methods, as well as the hard accuracy cliff imposed by the fixed number of sketch counters limit the practical effectiveness of these approaches. For example, on the realistic attack scenarios considered in § 8, the approximate methods in Jaqen and Euclid incur relatively high and unpredictable false-positive rates (up to ∼50%). Moreover, these proposals suffer sharp increases in FNR once the fixed resources allocated to the sketch are exhausted (*e.g.,* for a fixed-size LRU filter in Jaqen the FNR can go from less than 0.3% to more than 30% as the attack sources increases from 5k to 50K).
Machine-learning models for signature accuracy. Other recent efforts (*e.g.,* LUCID [24]) leverage machine learning (ML) techniques by casting detection of DDoS traffic as a classification problem [24], [46]–[50]. By using large vectors of traffic features computed for each potential attacker (*e.g.,* for each source address, for each flow), these efforts are able to produce highly accurate attack signatures thereby satisfying **R1**. However, such methods are currently not feasible to deploy in edge networks because of the excessive overheads of computing and communicating detailed features for large numbers of flows. For example the MAWILab traces used in § 8 contain millions of benign sources which would have to be classified.

Several recent efforts [25], [27], [51], [52] develop approaches to implementing trained flow-level models directly in switch hardware data planes which could be used to run ML signature detection at line rate. However, these approaches also face critical challenges to compute stateful features for large numbers (*e.g.,* millions) of flows in the already-confined switch hardware resources. For example, computing the features used in *ZAPDOS* for 1 M flows would require ∼80 MB of register memory which quickly surpasses the O(10 MB) of SRAM available in current programmable switch ASICs.[2]
Prefix-level refinement. To address **R2** and reduce the resources required for DDoS signature detection, a body of prior work considers prefix-level traffic monitoring (*e.g.,* [22], [37], [53]–[55]), but does not address the challenges imposed by modern attack methods. On the one hand, studies such as [37], [54], [55] identify the potential of prefix-level signatures to improve scalability, but their proposed algorithms are not robust against modern multi-vector and dynamic attacks (**R3**). On the other hand, recent efforts like ACC-Turbo [22] propose using switch hardware to accelerate refining prefixes, but the resource overheads

2. Note also that it is nontrivial to simply plug more SRAM into the switch pipeline due to power considerations.

they entail to handle large numbers of attack sources and attack vectors precludes their use in practice (*e.g.,* ALUs proportional to the number of prefixes). At the same time, these prior works only consider simple count-based features and struggle to compete with the accuracy of trained models. More generic and not DDoS-specific techniques for prefix-level iterative refinement are considered in studies such as [56]–[61], and we discuss their relevance and limitations for our work in § 5.

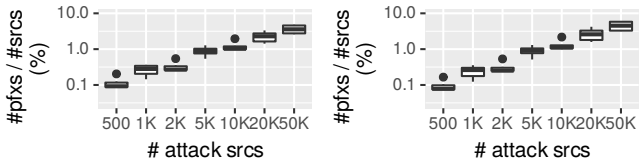
We note that several recent proposals use programmable switch hardware [23], [62], [63] to develop approaches to mitigating distributed link-flooding attacks [64], [65]. However, since these attacks target links in the network core rather than at the network edge and because the proposed methods require the participation of distributed switches across the network, these proposals address problems that are orthogonal to those considered by *ZAPDOS*.

2.4. Untapped Potential of Prefix-Level Signatures

Both sketch-based [20], [21], [38] and ML-based [24], [46], [47] approaches to detecting DDoS attack signatures are inherently limited by the tacit assumption that observed network source addresses are distributed uniformly at random across the IP space. For example, the use of pseudo-uniform hash functions in sketch-based approaches implies that approximation errors are uniformly distributed over the address space; that is, allocating more hardware resources has a uniform impact on all addresses. Similarly, collecting features and evaluating the same trained model across all addresses (or flows) in ML-based approaches also implies that the impact of monitoring resources on detected signature accuracy is the same for all addresses or flows.

For attackers, on the other hand, the costs of launching attacks from different parts of the address space are strongly non-uniform. Consider attackers who do not spoof their source addresses (*e.g.,* by using reflectors or directly flooding from bots). Prior work establishes that misconfigured hosts (which can serve as reflectors or bots) tend to cluster in prefixes associated with a few specific ASes [32], [33], [35], [66], [67] implying that attackers face higher costs to generate non-spoofed attacks from outside these prefixes due to the lower density of vulnerable hosts (see Appendix A for further justification). To quantify the impact of how non-uniform costs lead to clustering of malicious addresses, we determine the minimum number of prefixes required to exactly separate attack sources from the Mirai botnet [68] and benign sources from the MAWILab dataset [69]. We compute the ratio between this minimum number of prefixes and the total number of sources (attack and benign) for independent samples of up to 50k attack sources. Figure 2a shows that for the considered range of attack sources (x-axis), the minimum number of prefixes required is less than 5% of the total number of sources.[3]

Alternatively, consider attackers who spoof their source addresses. The most commonly considered approach to spoofing is to select attack sources uniformly at random from the set of all possible IP addresses [20]. However, Figure 2b shows that even under uniform spoofing, an optimal prefix-level signature still only requires monitoring a number of prefixes that is less than 5% of the number of sources to achieve perfect separation. Note that this is a result of the inherent clustering of benign addresses which can be formally described by a class of mathematical models known as multifractal measures [28]–[30]. Thus, to execute effective spoofed attacks against a suitably-designed prefix-level defense, attackers must invest additional cost to *infer* source addresses closer to benign traffic. However, since the cost for attackers to execute such inferences is largely unknown, in this work, we develop a practical method of generating attack sources which are *closer to* or *farther from* a given sample of benign traffic. We refer to this method as the *cost-based method* and formally introduce it in § 4.1.



(a) Mirai sources (b) Uniform random sources

Figure 2: Using prefix-level signatures can potentially satisfy both **R1** and **R2** because the # of prefixes required for *exact* separation of attack and benign sources is relatively low.

Thus, the main focus of our work is on fully unlocking the untapped potential of prefix-level volumetric DDoS signature detection by integrating sketch and ML methods with key methodological and algorithmic contributions. In particular, by using ML techniques instead of simple count-based thresholds, we improve the accuracy of prefix-level detection (**R1**) and by developing novel algorithmic components, we improve the refinement process against modern dynamic attack scenarios (**R3**), all the while ensuring high resource efficiency (**R2**).

3. Design and Overview of *ZAPDOS*

In this section, we first describe our threat model and then provide a brief overview of *ZAPDOS*, highlighting its key contributions and challenges.

3.1. Threat Model

Attacker's metrics of success. We assume a rational attacker whose goal during the active-attack phase is to inflict maximum damage (*i.e.,* loss) on the targeted network's benign traffic while minimizing the cost of launching the attack.

If the victim network has no defense, damage is measured in how much benign traffic is lost due to attack-induced congestion. Based on recent studies [33], [70], we assume the attacker uses one of several reflection attack vectors (e.g., DNS, NTP, SSDP) and/or botnets sending flooding attacks (e.g., SYN, ICMP, UDP). To launch these attacks the main cost for the attacker lies in acquiring sufficient attack sources (*e.g.,* bots) to maximize attack volume.

3. See § 4.1 for details of the method used to generate these scenarios.

If the victim network deploys signature-based defenses, the notion of damage as well as cost of attack is more complex. First, effective signature-based defenses increase the cost for the attacker since extra effort is required to evade detection. In particular, the attacker can combine several different attack vectors and change attack vectors and attack sources dynamically.[4] Second, signature-based defenses also introduce an additional type of damage in that the reported signatures may falsely include benign sources. The attacker can potentially leverage this by intelligently selecting attack sources to confuse or mislead signature detection. We assume the adversary uses one of three methods to select attack sources: *(i)* based on the actual IP address of the reflector[5] or bot; *(ii)* based on a fixed uniform random sample of the source address space; or *(iii)* based on proximity to benign traffic through the cost-based method. In the later case, we assume the adversary cannot guess exactly which sources will appear in benign traffic, but can pay a higher cost (in terms of effort during the pre-attack phase) to infer attack sources that have longer prefix overlap with benign sources.

Victim's metrics of success. The goal of the victim is to prevent as much attack-induced damage as possible, from either flooding loss or false-positive signatures. We assume the victim is able to detect when a volumetric DDoS attack occurs but does not have any additional information about the attack (*i.e.,* the victim has no prior knowledge of the attack vector or the attack sources). Anecdotal evidence suggests that for small to medium scale edge networks, volumetric DDoS attacks often cause spikes in packet and bit rates which can be detected using simple counters (*e.g.,* on the border switch in Figure 1). A variety of other volumetric DDoS occurrence metrics have been proposed and implemented on switch hardware [21], [31] and could be integrated with *ZAPDOS* . Finally, we assume the victim has the ability to deploy traffic monitoring for signature detection and the ability to mitigate detected attack traffic (*e.g.,* rate-limit, re-route through a scrubbing service).

Note that the assumptions made in our threat model about the type of attacks and methods for detecting attack occurrences are consistent with others (*e.g.,* [20]–[22]).

3.2. Overview of *ZAPDOS*

As shown in Figure 3, *ZAPDOS* is a closed-loop hybrid hardware-software approach to detecting prefix-level volumetric DDoS attack signatures. Operating during the active-attack phase, *ZAPDOS* uses a fixed set of *prefix monitoring slots* implemented in programmable switch hardware, control software, and a set of operator-defined mitigation policies. *ZAPDOS* uses the prefix monitoring slots to collect features for a fixed number of prefixes at a time, reacts to the results by updating which prefixes to monitor, and reports attack prefixes as soon as they are known to be separate from benign prefixes with sufficient confidence.

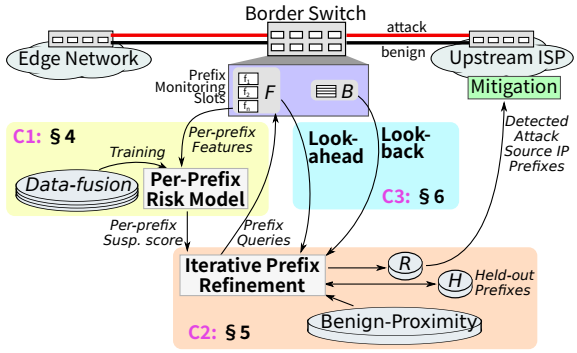


Figure 3: Overview of *ZAPDOS*.

The key contributions which enable *ZAPDOS* to satisfy **R1**-**R3** are concrete and novel solutions to the following challenges.

C1: How to determine if aggregate prefix-level traffic contains an attack source? The per-source or per-flow feature vectors used in prior ML-based approaches [24], [25] supply detailed signals directly correlated to the traffic entities to be classified. The per-prefix feature vectors considered in *ZAPDOS*, on the other hand, may contain signals from a variety of attack and benign sources and *ZAPDOS's* model must be able to "see through" these ambiguous prefixes in order to effectively refine and detect an accurate attack signature.

ZAPDOS trains a classification model over prefix-level features from a large number of scenarios with realistic packet-level attack traffic as well as attack and benign source address distributions, as described in § 4. In order to increase the model's ability to detect the presence of attack traffic in ambiguous prefixes, we develop a prefix length weighting method to explicitly bias the model's FNR vs. FPR tradeoff at different prefix lengths.

C2: How to use fixed switch resources to monitor a variable number of variable-length prefixes? The simplest iterative refinement approach simply zooms in or expands any prefixes that appear to contain any amount of volumetric DDoS attack traffic; however, this approach quickly leads to a large and infeasible number of prefixes to monitor. Since the total number and length of prefixes involved in an attack signature is unknown in the pre-attack phase, prior work which partitions the address space into a fixed number of prefixes (*e.g.,* DREAM [58]) is also insufficient.

ZAPDOS features a novel algorithm in § 5 to carefully decide which prefixes are worth monitoring and when particular prefixes can be reported as sourcing attack-only traffic with high confidence. The core idea of this algorithm is to partition prefixes as they are zoomed in on between a fixed-sized set of high priority prefixes to assign to monitoring slots and a variable-size set of less suspicious "held-out" prefixes. A recent snapshot of the structure of benign sources from the pre-attack phase is used to determine when a

4. Due to the overheads of precise clock synchronization (*e.g.,* access to GPS receivers) we assume a lower bound on how fast the adversary can coordinate these changes across their bots (*e.g.,* can only change attack parameters once every second).

5. Note that, although reflection attacks do require bots to spoof source addresses, the adversary is unable to spoof the addresses of reflectors which are observed at the victim network.

suspicious prefix has been zoomed in on enough to minimize false positives.

C3: How to ensure the refinement process remains robust when the adversary changes attack sources and vectors? A key limitation of iterative refinement approaches, including the base-line approach describe above, is that they assume the attack signature does not change rapidly. Modern DDoS attacks, on the other hand, may rapidly (e.g., every 30 s) change attack sources in order to confound such approaches.

We introduce two key algorithmic components in § 6 which allow the refinement process to quickly adapt to changes in the underlying attack traffic. First, *ZAPDOS* implements a "look-ahead" method to detect which child prefixes are active *before* allocating precious monitoring resources in the next epoch. Second, *ZAPDOS* implements a "look-back" method that keeps a low-overhead approximate record of which non-monitored prefixes were active to guide resource allocation in the next epoch when the attack sources change.

4. Per-Prefix Risk Model

At the heart of *ZAPDOS* lies a per-prefix risk model trained to report when a monitored prefix contains one or more volumetric DDoS attack sources.

Why use machine learning (ML)? To understand why we use an ML-based per-prefix risk model, consider a simpler method that decides if a prefix is suspicious based on applying a threshold to a single metric such as the difference between DNS requests and DNS responses as proposed at source-level in [20]. A key challenge with using a threshold in prefix-level detection is that the baseline volume of traffic changes drastically with prefix length and differently in different prefixes (*i.e.,* due to clustering [28]–[30]). A non-linear model trained on features from a diverse range of prefixes at different lengths as used in *ZAPDOS*, on the other hand, can capture the complex correlation between prefix length, traffic volume and other more descriptive features (*e.g.,* inter-packet gap statistics).

The challenge of appropriate data. As with other ML-based methods, the success of *ZAPDOS's* model depends first and foremost on the availability of a *large* and *representative* dataset of observed feature vectors and labels from both classes. However, limited availability of appropriate data is a persistent and well-established problem when applying ML classification techniques to network security tasks [71], [72]. In developing *ZAPDOS*, this problem is particularly pronounced because the model must be trained over observations that capture the realistic blending of attack and benign features under prefix aggregation. Datasets and methodologies used in prior efforts are insufficient because they either contain no benign traffic [21], [35], [73], lack high-confidence labels [70], [74], or do not provide realistic address-space distributions [20], [75]. To illustrate, consider for example the CIC datasets [75] used to both train and evaluate several proposed approaches to classifying volumetric DDoS attack flows [24], [46]. When viewed at the five-tuple flow level, these datasets are quite large, containing millions of flows. However, as shown in Table 2, when viewed at the source address or source address prefix level, their size quickly collapses to an extremely small number of distinct observations. The largest (ISCX'12) contains 14 attack sources distributed in 6 /16 prefixes.

Dataset	# Benign			# Attack		
	/16	/24	/32	/16	/24	/32
CAIDA'07 [73]	0	0	0	4 k	8.7 k	9 k
ISCX'12 [76]	1590	2041	2129	6	9	14
CIC'17 [75]	922	2125	3432	1	1	1
CSECIC'18 [77]	1	6	446	4	10	10
Ours[6]	30 k	3.2 m	4.8 m	7 k	45 k	50 k

TABLE 2: Number of distinct attack and benign prefixes in datasets used for training and testing in LUCID [24].

4.1. Data-fusion for Realistic Prefix-level Features

Given the insufficiency of existing datasets, we develop a novel data-fusion approach to training and evaluating *ZAPDOS*. As shown in Figure 4, our method involves several real-life and synthetic data sources which provide realistic attack address distributions as well as packet-level data. This method allows us to generate multiple, independent attack scenarios, each with realistic, disjoint sets of attack and benign sources and hence representative blending of features under prefix aggregation. In particular, we take extra caution to ensure proper separation of prefix-level features between training and testing sets to avoid model overfitting. Overall we generate 84 distinct attacks with varying numbers of sources and attack vectors, 42 training scenarios and 42 test scenarios, using the following steps.

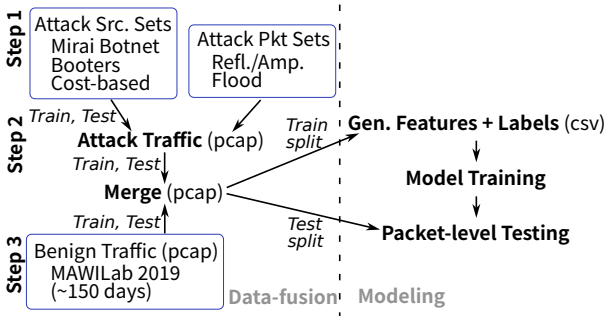


Figure 4: Diagram of dataset preparation and modeling methodology used in *ZAPDOS*.

Step 1: Generate attack source sets. We use three methods of obtaining realistic sets of attack source addresses: Mirai botnet [68] (∼18 M distinct sources), Booters dataset [35] (∼18 K distinct sources), and a synthetic cost-based method. For our training data, we extract disjoint sets with varying numbers of sources from the Mirai data set (for simplicity we use sets of 500, 1k, 2k, 5k, 10k, 20k, and 50k sources). Booters sources are used in our evaluation to demonstrate how *ZAPDOS* can generalize to entirely unseen source distributions.

6. In particular, counts from one training attack scenario with 50k attack sources sampled from the Mirai dataset [68] and benign traffic from MAWILab [69] constructed using method described in § 4.1.

We also develop a *cost-based method* to understand how *ZAPDOS* reacts when the adversary invests effort to infer benign prefixes during the pre-attack phase and spoofs attack sources to come from these prefixes. In particular, we define cost for the attacker in terms of how many prefix bits ℓ attack sources share with benign sources. Then, given a set of benign sources, to generate attack sources for a particular cost ℓ, we identify the set of $/\ell$ prefixes that share a $/(\ell - 1)$ prefix with a benign source, then select attack sources uniformly at random in these identified prefixes. In this way the clustering of attack sources is closer to (farther from) benign sources for larger (smaller) values of ℓ.

Step 2: Generate attack traffic. Next, we generate packet-level attack traffic for each enumerated attack source based on attack vectors. Table 3 shows the six different attack vectors as well as packet-level sources or parameters considered in this work. We fix the target bit rate of the aggregate attack traffic (*e.g.,* 1 Gbps) and the number of attack sources to compute the per-source bit rate which determines the arrival time of attack packets. Here, an attack *A1* with the same target bit rate as *A2*, but with more attack sources, will have lower per-source bit rates (and vice versa). To mimic a real attack [35], each source generates packets at a constant bit rate with a uniform $\pm 1\%$ variance around the target per-source rate.

Type	Attack Vector	Packet Source/Type
Refl./Amp.	DNS	"Booter1" [35]
	NTP	CIC DDoS "day two" [78]
	SSDP	Sucuri analysis [79]
Flood	SYN	TCP with SYN flag set
	ICMP	ICMP Echo request
	UDP	UDP random payload

TABLE 3: Attack vectors generated for evaluation of *ZAPDOS* and the sources or types of attack packets used.

Step 3: Merge attack and benign traffic. Finally, we interleave the packets of each generated attack with benign traffic from the 2019 MAWILab dataset [69] (preserving the relative packet arrival times of both traces). The MAWILab dataset provides packet-level data for a large number of benign traffic excerpts allowing us to assign a unique excerpt to each generated attack scenario. Although using a single supplier of benign traffic does not allow us to reason about a trained model's ability to generalize to other network settings, our data-fusion methodology can be replicated with any other source of benign traffic in order to produce datasets and models tailored for particular networks.

4.2. Modeling Setup

Feature selection. To train the classification model used in *ZAPDOS*, we compute the features described in Table 4 for each prefix in each of the training attack scenarios. At a high level, we carefully selected features that on the one hand intuitively capture differences in traffic patterns between attack and benign behaviours and on the other can be computed in switch hardware at line rate. Though similar features have been used for flow-level attack classification [24], [50], [80], [81], we reiterate that in *ZAPDOS*, all features are aggregate over all traffic from the prefix being classified. As a result, we only require maintaining feature state during each epoch for a small, limited number of prefixes using queries described in § 5 instead of over all flows as in prior efforts.

	Feature	Description
	Prefix length	Number of leading bits defining prefix.
Directional	Bytes from	Number of bytes from this prefix.
	Bytes to	Number of bytes to this prefix.
	Packets from	Number of packets from this prefix.
	Packets to	Number of packets to this prefix.
Statistical	min/max/ave. len.	Minimum, maximum, and moving average of length of packets from or to this prefix.
	min/max/ave. IPG	Minimum, maximum, and moving average of time between consecutive packets from or to this prefix.
	Last active	Number of epochs since any packets were last observed from or to this prefix.
Generic	$rrDiff$	Maximum (responses - requests) over a number of known amplification protocols (e.g., DNS, NTP, etc.), see Appendix B.

TABLE 4: List of features computed each epoch for each monitored prefix for use in *ZAPDOS*'s model.

Model selection. We use a random forest (RF) model [82], [83] with 500 trees and other defaults as set in [84] to predict if a prefix sources attack traffic based on the observed features. We chose to use RF due to its observed high accuracy and fast training and classification times compared to more complex methods like neural networks.

Weighted training. Initial experiments suggested that the per-prefix false-negative and false-positive rates of our model do not directly correspond to the error rates of *ZAPDOS's* end-to-end iterative refinement approach. In particular, we found that false-negative decisions made at shorter prefix lengths had a disproportional impact on the overall false-negative rate because they caused the refinement process to miss large prefixes potentially containing many attack sources. To counteract this effect, when training our model, we use a weighted sample of our training set which contains more observations of attack prefixes at shorter prefix lengths (*e.g.,* less than /16) compared to longer prefix lengths. This weighting of the training set causes the model to be more suspicious (*i.e.,* make more false-positive decisions) at shorter prefix lengths, making it more likely that *ZAPDOS* will continue zooming in and not miss large regions that include attack sources.

5. Iterative Prefix Refinement

5.1. Baseline Approaches

In general, approaches to prefix-level iterative refinement [57], [58] maintain a list of monitored prefixes F. Packets are grouped by these prefixes and aggregate features computed for each prefix (*e.g.,* by submitting dataflow queries [31], [85], [86]) during a monitoring window or *epoch* (*e.g.,* 1 s). Between epochs a set of decisions are made about how to update F for the next epoch based on features computed in the previous epoch. Typically these decisions are to "zoom-in" on a particular prefix (*e.g.,* replace 10.10.0.0/16 with 10.10.0.0/17 and 10.10.8.0/17) or to "zoom-out" by combining two sibling prefixes. In the case

of volumetric DDoS signature detection, the goal is to zoom-in on prefixes that contain attack sources and zoom-out on benign prefixes.

Existing approaches to prefix-based refinement have one of two undesirable properties. First, approaches like MRT [57] use tree-like data structures to implement prefix matching on CPUs and zoom-in on every suspected attack prefix, leading to an exponential increase in the number of prefixes whose features must be collected and processed during each epoch. Second, approaches like DREAM [58] use a fixed number of TCAM entries on programmable switches and every time they zoom in on a prefix, they also choose another pair of siblings to zoom out on.

Our initial experiments quickly confirmed that neither of these approaches are sufficient for volumetric DDoS attack signature detection. The MRT approach of zooming in on every prefix classified as suspicious quickly exhausts the limited number of TCAM slots. The proposed algorithms in DREAM [58] also struggle to effectively zoom in to sufficiently long prefix lengths. Consider the simple example where there is only one attack prefix, 10.0.0.0/8. As shown in Figure 5, DREAM requires monitoring eight other prefixes even though they may be entirely empty.

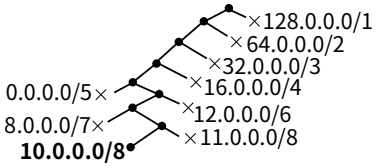


Figure 5: Monitoring one attack prefix 10.0.0.0/8 in DREAM [58] requires monitoring an additional 8 benign or empty prefixes in order to maintain a complete cover of the address space.

5.2. Scheduling Prefixes on Fixed Monitoring Slots

To address these algorithmic short-comings in the light of volumetric DDoS attack signature detection, in *ZAPDOS* we develop a novel approach based on the high-level idea of scheduling which prefixes to monitor each epoch. We augment the set of prefixes F with two other sets: R which contains prefixes which have already been reported as part of the attack signature, and H which contains non-scheduled prefixes "held-out" in CPU memory for consideration in future epochs. During each epoch, we use a fixed number of prefix monitoring slots compiled into switch hardware to compute features for prefixes in F (each slot computes all features for a single prefix). Between epochs, we apply a procedure called EPOCHUPDATE() to update all three sets to zoom in on attack signatures. The key decisions of our approach stem from two high-level scheduling policies, *children-first* and *never-zoom-out* as illustrated in Figure 6.

Children-first. The children-first policy is based on the intuition that if the model classifies a prefix as containing attack traffic, it is more likely that a child of that prefix will also contain attack traffic in the next epoch compared to any other prefix. As a result, we schedule all newly-zoomed-in-on children before any other prefixes under consideration.

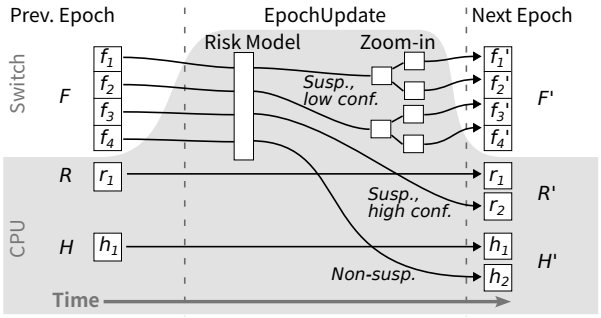


Figure 6: Iterative prefix refinement to zoom-in on suspicious prefixes while using fixed monitoring resources.

Only once these child prefixes have been cleared as non-suspicious, do we remove them from F to H.

Never-zoom-out. The never-zoom-out policy is based on the intuition that even if a prefix does not look suspicious in a particular epoch, it may still become suspicious in the future (*e.g.,* due to changes in attack source, or fluctuations in benign traffic). As a result, instead of zooming out as in DREAM, we collect all prefixes ever zoomed in on in H. This way, even if the attack vectors or sources change, *ZAPDOS* can quickly recall its previous progress and continue zooming in where it left off, thus avoiding the slow control loop vulnerability [22] associated with approaches that do not maintain this kind of longer-term state (*e.g.,* Jaqen [20]).

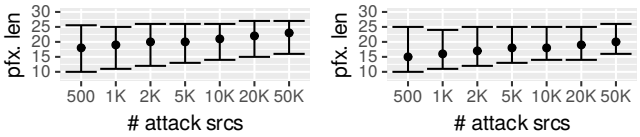
Note that although F is constant in size, $R \cup H$ grows after most epochs during the active-attack phase. We leave questions of how to interpret and ultimately reset F, R, and H in the post-attack phase and/or how to optimally precondition these sets in the pre-attack phase to future work.

Key parameters. *ZAPDOS* includes two key parameters that effect the iterative refinement process. First, `prefixesPerEpoch` determines the number of feature monitoring slots available in switch hardware (*i.e.,* an upper bound on $|F|$). Increasing `prefixesPerEpoch` enables *ZAPDOS* to observe a larger region of the address space each epoch and can in some cases reduce the number of epochs to reach an accurate attack signature, but increases *ZAPDOS's* hardware resource footprint. Second, `bitsPerEpoch` determines how many bits are added to the length of prefixes when *ZAPDOS* zooms in on them. Although intuitively larger value of `bitsPerEpoch` allow *ZAPDOS* to zoom-in faster, since each zoom-in decision generates `2^bitsPerEpoch` children, setting `bitsPerEpoch` too high generates too many children to monitor and can actually cause *ZAPDOS* to zoom-in slower due to the limit imposed by `prefixesPerEpoch`.

5.3. Deciding Length of Reported Prefixes

Given that prefix lengths increase monotonically in *ZAPDOS* (by the never-zoom-out policy), a key consideration is when a prefix flagged by the model as potentially containing attack traffic should be reported in the attack signature. On the one hand, reporting at too short of prefix

lengths leads to high false-positives. On the other hand, reporting at too long of prefix lengths leads to wasting monitoring resources and increase detection time. Ultimately, as shown in Figure 7 optimal prefix-level partitions of attack and benign sources require a wide range of prefix lengths (*e.g.,* /10 to /25).



(a) Mirai sources (b) Uniform random sources

Figure 7: Distribution of lengths of prefixes used for optimal separation between attack and benign traffic from Figure 2.

To better approximate these optimal prefix-level partitions, *ZAPDOS* develops an "early stopping" method to decide when prefixes that are flagged by the model should be included in the report set R (and hence removed from active consideration in F and H). Since the primary concern is to avoid reporting prefixes that also contain benign sources, *ZAPDOS's* early stopping leverages a profile of benign traffic collected during the pre-attack phase.

In particular, we define $\textsc{Benign-Proximity}(y, p) = n/2^{32-p}$ for a given prefix y of length p where n is the number of benign sources observed in y during the pre-attack phase. During the active-attack phase, if the model flags a prefix f of length ℓ_f as suspicious, we report f and move it to R if and only if $\textsc{Benign-Proximity}(f, \ell_f)$ is less than a threshold. Computing $\textsc{Benign-Proximity}(y, p)$ requires monitoring distinct sources for the most recent m seconds in the pre-attack phase. In our evaluation, we found that m=120 s yields sufficiently accurate results while requiring modest resources. For example, in our hardware prototype we used a ∼131 KB Bloom filter.

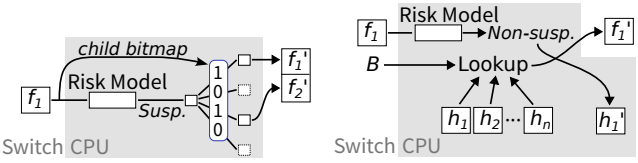
6. Tuning Refinement for Dynamic Attacks

Although the children-first and never-zoom-out policies are key to realizing *ZAPDOS*, they require additional algorithmic components (beyond those described in § 5) to cope with dynamic changes to attack sources observed in modern volumetric DDoS attacks.

6.1. Look-Ahead to Avoid Empty Children

The children-first policy requires allocating monitoring slots for all children of any prefix flagged by the model as suspicious but not yet ready for reporting. However, due to the relative sparsity of observed addresses, some of these children almost always turn out to be inactive and hence waste precious switch resources. For example, with `bitsPerEpoch` = 4 and a parent prefix with only one active child, *ZAPDOS* would waste 15/16 monitoring slots.

To address this problem, we develop a novel *look-ahead* method that encodes information about which children are active in the features gathered for each parent prefix. In particular, as shown in Figure 8a we add a per-prefix "child bitmap" where each bit represents a potentially active child. When a packet matches prefix f of length ℓ_f, we extract the ℓ_f + `bitsPerEpoch` bits of that packet's source address and use them as an index into f's child bitmap. During EpochUpdate(), if f is identified as suspicious by the model, we then read f's child bitmap and only select the children of f whose bits were set for monitoring in the next epoch. The child bitmap requires adding an extra feature with 2^`bitsPerEpoch` bits per prefix. For example, at `bitsPerEpoch` = 4, this only adds 16 bits.



(a) Look-ahead increases zoom-in rate by focusing on active children.

(b) Look-back brings back active hold-out prefixes when attack changes.

Figure 8: Extensions to the scheme of Figure 6 to deal with dynamic changes in the set of attack sources.

6.2. Look-Back to Catch Changes

Although the never-zoom-out policy ensures that *ZAPDOS* can always make progress towards refining attack signatures even when the attack sources change, a critical consideration not addressed in § 5 is which prefixes from H should be added to F when extra monitoring slots are available. Again, due to relatively sparse population of the observed address space, simplistic methods like taking the first `prefixesPerEpoch` from H lead to wasting monitoring slots on empty prefixes.

To address this problem, we develop a *look-back* method that casts a wide net over all regions of the address space not monitored in F. Our key observation is that we do not necessarily need to discover exact regions sourcing new attack traffic, but only need to re-focus the refinement process on currently active regions of the address space. In particular, as shown in Figure 8b we use a simple Bloom filter [87] B to build an (approximate) list of distinct sources that don't match prefixes in F. Then when extra monitoring slots are available, we select prefixes from H based on their membership in B.

7. *ZAPDOS* Prototype

We prototype *ZAPDOS* using a Tofino-enabled Wedge 100BF-32QS switch[7]. The switch ASIC runs a P4 [88] program that computes per-prefix features in hardware registers (Table 4), maintains look-ahead and look-back components (§ 6), and classifies packets associated with reported attack prefixes in R. Table 5 shows that *ZAPDOS* has a relatively small footprint compared to the available hardware resources for key resource types like SRAM and TCAM, comparable to the footprint of Jaqen[8].

7. https://www.edge-core.com/productsInfo.php?cls=1&cls2=5&cls3=181&id=770

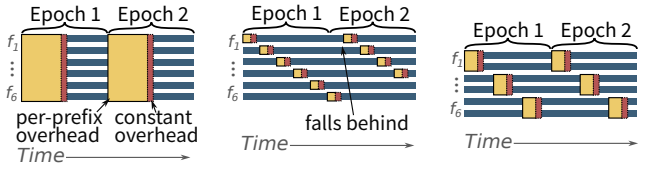
8. Note that Jaqen only reports hardware resource usage of their detection module, but their mitigation modules, which actually generate source-level attack signatures, require unknown additional resources.

Crossbar	SRAM	TCAM	VLIW	Hash Bits	ALU	Gateways
6.70 %	7.29%	11.11%	9.38%	17.11%	26.04%	15.62%

TABLE 5: Switch hardware resources used by the *ZAPDOS* data plane as percentage of total available on Tofino.

The switch CPU runs the *ZAPDOS* control plane, a Haskell [89] program which evaluates the per-prefix classification model (§ 4) and refinement algorithm (§ 5), interacting with the ASIC through the `bfruntime` gRPC interface and ASIC's CPU port. A key challenge in prototyping *ZAPDOS* in hardware is optimizing the communication between ASIC registers and the switch's CPU to minimize the update time overhead. In addition to carefully multi-threading the *ZAPDOS* control plane, we leverage two other techniques to optimize this communication: *(i)* batch-round-robin epochs and *(ii)* packet ferries.

Batch-Round-Robin Epochs. Naïvely reading the features of all prefix monitoring slots in one shot at the end of each epoch (as shown in Figure 9a) requires a significant amount of time. For example, writing a new set of 1500 prefixes to monitor in hardware takes ∼0.5 s, implying traffic would only be monitored half of the time with 1-second epochs. On the other hand, incremental updates that cycle through prefixes in round-robin order (as shown in Figure 9b) cause the iterative refinement process to fall behind due to the constant overhead associated with each RPC. For example, an RPC updating a single prefix takes ∼5 ms, implying one cycle through all 1500 prefixes would take ∼7.5 s.



(a) Sync. Epochs (b) Round-Robin (c) Batch-Round-Robin

Figure 9: Illustration of different possible approaches to updating prefix monitoring slots in *ZAPDOS*.

In *ZAPDOS*, we develop an approach called *batch-round-robin* which updates batches of prefixes in a single RPC and cycles through batches in round-robin order as shown in Figure 9c. This combines the benefits of synchronous epochs and round-robin updates because the constant overhead of each RPC is amortized over all prefixes in a batch and each batch is much faster to update compared to the entire set of monitoring slots. Updating a batch of 100 prefixes in our prototype takes ∼25 ms so the 15 batches required to update all 1500 prefixes takes 375 ms implying batch-round-robin can easily keep up with 1-second epochs.

Packet Ferries for Feature Collection. To implement the modeling and iterative refinement methods, the *ZAPDOS* control plane needs to read features of all 1500 prefix monitoring slots back to the CPU each epoch, but a naïve batch-read RPC request takes ∼1.3 s. Instead, *ZAPDOS* implements a technique called *packet ferries*, inspired by in-band network telemetry [90], [91] and further developed in [92]. As shown in Figure 10, packet ferries send specially-marked request packets through the *ZAPDOS* data plane using the switch CPU's backplane port which bypasses gRPC and driver layers allowing *ZAPDOS* to read all 1500 prefixes in ∼50 ms.

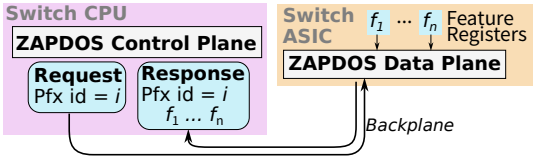


Figure 10: Packet ferries enable fast reads by bypassing the gRPC and driver layers.

8. Evaluation

We build a large dataset of realistic attack traces using the methodology described in § 4.1 and use this dataset to evaluate *ZAPDOS*. We show that *ZAPDOS*

- quickly detects attack signatures with high accuracy in our hardware prototype implementation and we provide a breakdown of the latency overhead (§ 8.2),
- accurately detects signatures of modern multiple-vector and dynamic attacks with a single model (§ 8.3),
- maintains low error rates for changes to attack parameters (*e.g.,* numbers of sources), system parameters (*e.g.,* `prefixesPerEpoch`), and loss scenarios (*e.g.,* flooded border links) (§ 8.4), and
- is robust against attackers who can spoof attack traffic to come from the same prefixes as benign traffic maintaining orders of magnitude lower error rates compared to prior approaches when given comparable traffic monitoring resources (§ 8.5).

8.1. Setup

Success metrics. The goal of *ZAPDOS's* attack signatures are to effectively reduce the amount of damage caused by an attack regardless of how much effort the adversary spends on the attack as described in § 3.1. To measure potential damage, we measure the accuracy of given attack signatures w.r.t. ground-truth in terms of per-byte *false positive* (FPR) and *false negative* (FNR) rates. Smaller FPR and FNR indicate more accurate attack signatures and in turn less damage for the target network. To measure how fast an attack is detected, we use the *detection time* metric that we define to be the time difference between when an attack starts and when *ZAPDOS* reports more than a certain fraction of the attack prefixes. Note that we classify each packet as a false-positive or false-negative based on the current value of R when that packet arrived and compute aggregate FPR and FNR over the entire attack scenario. As a result, these metrics also reflect how quickly *ZAPDOS* was able to refine accurate attack signatures (*e.g.,* if *ZAPDOS* takes longer to detect a particular attack prefix, it will make a larger contribution to FNR). The shown error bars mark the 5^{th} percentile, the median, and the 95^{th} percentile, respectively over independent trials.

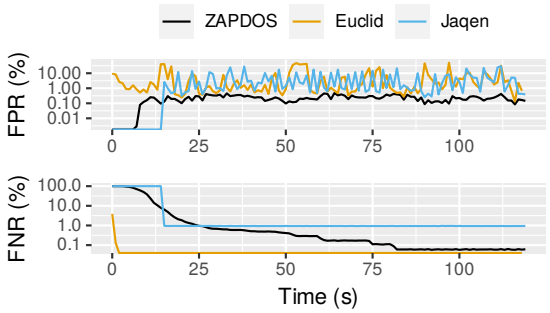
Other DDoS defense approaches. We compare *ZAPDOS* with two state-of-the-art switch-based approaches to volumetric DDoS defense in edge networks: *(i)* Euclid [21] uses a sketch-based method to detect attack sources by estimating their contributions towards the increase in the entropy associated with the attack, and *(ii)* Jaqen [20] is a library of sketch-based detection and mitigation primitives that can be deployed on switch hardware with a CPU-based controller. We use an exact heavy-hitter computation to represent the best-case scenario for Jaqen, which originally used universal sketching to estimate heavy hitters. We do not compare with approaches like Poseidon [38] as it focuses on local solutions with no clear separation of detection and mitigation. We also do not compare with ML approaches like LUCID [24] since their feature-gathering overheads are infeasible for edge networks as described in § 2.3.

Default parameters. As discussed in § 5 and § 6, the two main parameters controlling efficiency of iterative refinement in *ZAPDOS* are `prefixesPerEpoch`, and `zoomInBits`. Unless otherwise noted, we set `prefixesPerEpoch` to 1500 and `zoomInBits` to 4 since these values yielded acceptably low error rates in initial experiments and acceptably low overheads in our hardware prototype. We use a Bloom filter with 2^{20} bits and a single hash function in our prototype for look-back. We set a default epoch duration of one second. The pre-attack phase lasts for 120 s to allow *ZAPDOS* to compute the benign traffic profile described in § 5.3 and we set the benign proximity threshold to 0. Unless otherwise noted, the active-attack phase lasts for 120 s with an aggregate attack rate of 1 Gbps (enough to flood an access link of a small or medium-sized campus network) using 50k attack sources from Mirai data [68] (distinct from the training sources) comparable in size to real-world volumetric DDoS attacks [35].

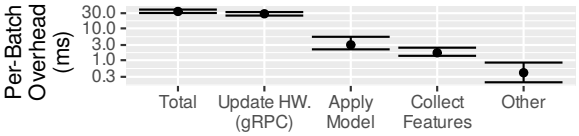
8.2. *ZAPDOS* Prototype Performance

To evaluate our prototype, we replay an independent UDP-flood attack trace from the test partition of our dataset containing a mix of attack and benign traffic.[9] The prototype is switched from collecting the benign traffic profile (§ 5.3) to actively zooming in on attack prefixes when the trace enters the active-attack phase at 120 s. We collect the list of reported attack prefixes in a file which also includes a timestamp of when each prefix was reported, then compute per-byte FPR and FNR in each 1 s time window based on the timing of these reports offset relative to the original trace.

Figure 11a shows the evolution of FPR and FNR in our prototype over the duration of the attack in comparison with simulated implementations of Euclid and Jaqen on the same trace with similar resource budgets. In *ZAPDOS*, FNR quickly drops below 1% after 25 s (going down to ∼0.05% by 120 s) as more attack prefixes are correctly reported. FPR remains low and stable around 0.2%. In comparison, Jaqen deploys mitigation after 15 s due to overheads of installing



(a) Timeseries of *ZAPDOS* prototype accuracy compared with simulation results of Euclid and Jaqen on the same trace with comparable resource budgets.



(b) Characterization of the batch-round-robin update time overhead with 100 prefixes per batch.

Figure 11: Performance of our *ZAPDOS* prototype on a realistic attack scenario.

the particular mitigation module for UDP flood attacks, then achieves higher FPR (∼4%) and FNR (∼1%). Euclid's pure data plane approach begins mitigating attack traffic within 1 s and identifies nearly all attack sources, but also produces erratic FPR (up to 50%) indicating significant impact on benign traffic. This result demonstrates that *ZAPDOS* not only effectively detects volumetric DDoS attack signatures in switch hardware, but also consistently achieves lower error rates compared to best-case simulation performance of Jaqen and Euclid.

We also measured the median absolute difference in accuracy between our *ZAPDOS* prototype and *ZAPDOS* simulator over all epochs considered in Figure 11a to be ∼4.1e-04% for FPR and ∼0.17% for FNR. We attribute the slightly larger gap in FNR to inaccuracies outside of *ZAPDOS* in the tool used to replay attack and benign attack traffic.[10] Given this close correspondence between prototype and simulator, in the rest of this section we show results from our simulation since it enables higher-confidence accuracy computation and evaluation of multiple traces in parallel.

Finally, we measure the latency overhead of our batch-round-robin method (§ 7) with 100 prefixes per batch. Figure 11b shows the median total per-batch latency overhead over all epochs of the attack scenario as well as break down across three main operations. Latency overhead is clearly dominated by hardware update time which could be optimized using, for example, DMA rather than gRPC.

9. In particular, we use cost-based attack sources with attack cost ℓ = 16 combined with MAWILab 2019-02-05 benign traffic.

10. In particular we use `tcpreplay` which appears to fall behind realtime during the active-attack phase leading to delayed timestamps from the prototype which inflate FNR.

8.3. Performance Against Modern Attacks

8.3.1. Simple attacks: Single-vector and static. Modern volumetric DDoS attacks leverage diverse vectors to generate large volumes (*i.e.,* data rates) of attack traffic toward their victims. To demonstrate *ZAPDOS's* ability to identify attack sources regardless of the attack vector used, we generate an independent single-attack scenario for each of the attack vectors described in Table 3. For each attack we draw a sample of 50k distinct attack sources from the Mirai [68] dataset, the Booters [35] dataset, or "spoofed" sources from a uniform random distribution (Rnd.). Due to iterative refinement in *ZAPDOS*, we report in Figure 12 FPR and FNR of the attack signature detected by *ZAPDOS* for each attack vector at three different points in time after the start of the attack. For all attacks, FNR decreases down to ∼0.1% after 60 s as *ZAPDOS* reports more attack prefixes. FPR, on the other hand, does not follow a clear trend beyond stabilizing between 0.1% and 0.3% for all attacks. We attribute this to the inherent burstiness of benign traffic which causes burstiness in the FPR time series (*e.g.,* see Figure 11a). Practically, this result demonstrates that the single trained model in *ZAPDOS* is able to generate highly-accurate attack signatures for a wide range of modern volumetric DDoS vectors within a couple minutes of the onset of that attack.

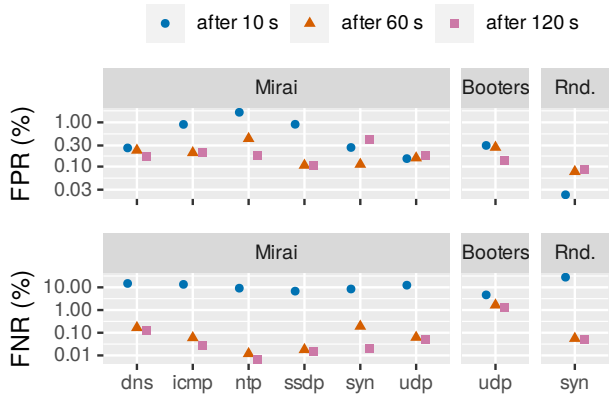


Figure 12: Performance of *ZAPDOS* on different single-vector, static attacks with 50k distinct sources.

8.3.2. Complex attacks: Multi-vector and dynamic. In addition to using diverse attack vectors individually, modern DDoS attacks are also known to combine multiple concurrent attack vectors and change them over time [20], [23], [38], [64]. To demonstrate *ZAPDOS's* ability to handle such complex attacks, we generate multi-vector dynamic attack scenarios by selecting nine distinct attack-vectors from Table 3 and attack source sets from the Mirai dataset (using the discrete set sizes described in § 4.1). During the attack scenarios, at regular intervals the attacker randomly selects three attack vector, attack source set pairs from these nine and combines the traffic of all three against the victim. We generate several such scenarios with different intervals between attack changes, in particular 1 s (fast), 5 s (medium), and 10 s (slow). Figure 13 shows the evolution of attack signatures generated by *ZAPDOS* during the first 120 s of each scenario. *ZAPDOS* is relatively slower to refine attack signatures compared to previous experiments because these attacks contain roughly three times the number of distinct sources. However, we note that the rate of signature refinement remains the same regardless of how fast the attack changes between attack sets. This illustrates the effectiveness of both "look-ahead" and "look-back" methods (§ 6) to quickly focus limited monitoring resource on target prefixes and to switch prefixes when the underlying attack changes.

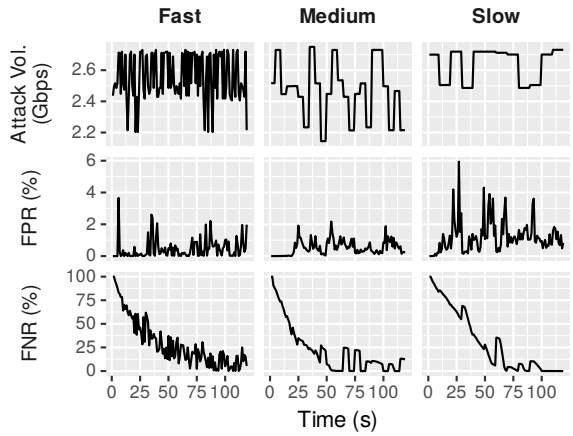


Figure 13: Performance of *ZAPDOS* in refining signatures of multi-vector, dynamic attacks with varying numbers of sources.

8.4. Sensitivity Analysis

To understand how *ZAPDOS* performs on a wider range of parameter settings and attack scenarios, we evaluate the impact of changing both the resources allocated to *ZAPDOS* (*i.e.,* `prefixesPerEpoch`) and attack parameters (*i.e.,* the number of distinct attack sources at a fixed aggregate attack rate). We also investigate the robustness of *ZAPDOS* in scenarios where the edge network's border link is flooded. Due to space limitations, we describe the details of these experiments in Appendix C and summarise the key takeaways here. *ZAPDOS* is able to leverage larger switch hardware state (via larger `prefixesPerEpoch`) to reduce the false negative rate and detection time. On the other hand, although larger numbers of sources sending a fixed overall attack rate presents a harder challenge, for 1 Gbps attacks with up to 50 k sources *ZAPDOS* is still able to achieve FPR $< 0.5\%$, FNR $< 5\%$, and to report 90% of attack sources within 10 s. Finally, *ZAPDOS* is robust against changes to feature distributions caused by a flooded border link and reduces loss-induced damage to below 10% in less than 10 s on average.

8.5. Impact of Proximity of Attack Sources

A fundamental concern with prefix-level attack signatures as detected in *ZAPDOS* compared with source-level signatures as detected in Euclid and Jaqen is that a resourceful adversary could spoof attack sources to fall into known

benign prefixes thereby triggering high false-positive rates.[11] We evaluate this concern by generating 10 independent UDP flooding attack scenarios. Each scenario uses cost-based attack sources as described in § 4.1 at four different settings of cost parameter ℓ and distinct benign traffic from MAWILab. Recall that higher-cost attack sources are chosen to share longer prefixes with benign traffic (*e.g.,* for cost $\ell = 24$ all attack sources share a /24 prefix with at least one benign source). We use these scenarios to empirically quantify the impact of cases where attack sources are intentionally crafted to induce high FPR and to compare this impact with performance of Euclid and Jaqen on the same traces using the same methodology as in § 8.2 above.

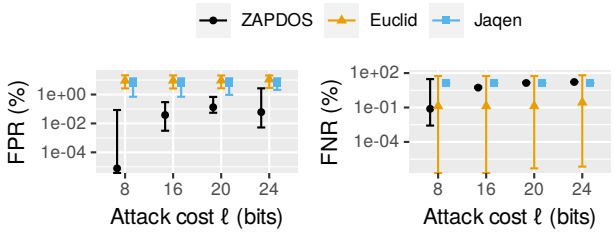


Figure 14: Aggregate performance comparison over 10 independent attacks. Higher attack cost ℓ indicates longer prefix overlap between attack and benign sources.

Figure 14 shows FPR and FNR over all 10 attacks at each setting of cost parameter ℓ. Performance of *ZAPDOS* does depends on attack cost, with cheaper attacks yielding lower error rates (*e.g.,* median FPR of $\sim 8 \cdot 10^{-6}\%$ at $\ell = 8$ bits compared to 0.06% at $\ell = 24$ bits). However, we note that this trend is concave-down and there is no apparent breakdown of *ZAPDOS's* prefix-level signatures even up to $\ell = 24$. Although the FPR of Euclid and Jaqen are not impacted by attack cost, they are *orders of magnitude* higher (∼9% and ∼6% respectively) compared to *ZAPDOS's* prefix-level signatures.

Interestingly, we observe that attack cost also has an impact on *ZAPDOS's* FNR which ranges from ∼0.07% to ∼17% as ℓ increases. This is a result of longer detection time required by higher-cost attacks as discussed below. In comparison, Jaqen achieves relatively constant FNR around 13% and Euclid's FNR is highly dependent on characteristics (*e.g.,* entropy) of each trace though uncorrelated with attack cost.

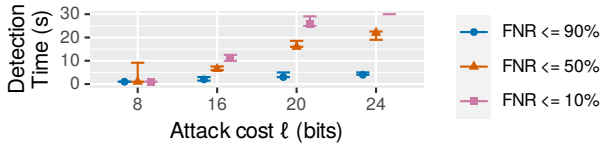


Figure 15: Detection time of *ZAPDOS's* attack signature coverage.

11. Note that since Euclid and many of the mitigation primitives in Jaqen are still source-level, cases where attack and benign traffic comes from the same source, *e.g.,* due to NAT, are a common problem across all these methods.

Intuitively, due to *ZAPDOS's* benign-proximity method for deciding the length of reported attack prefixes (see § 5.3), when attack sources are closer to benign sources, *ZAPDOS* must zoom-in to longer prefix lengths requiring more iterations. Figure 15 quantifies this effect by showing the time between the beginning of the attack and when *ZAPDOS's* FNR line falls below 10%, 50%, and 90%—in other words, the *detection time* from when the attack starts until when *ZAPDOS's* attack signature matches 10%, 50%, and 90% of all attack traffic each time window. For the lowest cost attacks *ZAPDOS* detects 90% of attack traffic within the first 1 s epoch. However, since for higher-cost attacks, attack and benign sources share longer prefixes, *ZAPDOS* takes longer (up to 24 s in the worst case) to produce sufficiently refined attack signatures. The key takeaway is that if the attacker expends extra effort to place attack sources closer to benign sources, *ZAPDOS* does not falsely block benign sources, but drills down deeper into the prefix tree to maintain low FPR at the cost of increased signature detection time.

9. Adversarial Considerations

In addition to attackers who intelligently spoof source addresses to fall within benign prefixes as considered in § 8.5, several other aspects of *ZAPDOS* could potentially result in vulnerabilities which attackers could exploit to thwart detection. We raise several possibilities and note that the severity of each depends on details of a production deployment of *ZAPDOS*.

Securing communication in *ZAPDOS*. Since *ZAPDOS* uses the switch data plane to collect feature results, an attacker could potentially spoof these request or response packets and jeopardize the accuracy of *ZAPDOS's* features. *ZAPDOS's* data plane could be modified to include distinct signatures on all result packets (*e.g.,* adding a secret written only through the PCIe channel) allowing the *ZAPDOS* control plane to verify all received feature results.

Timing-based attacks. A skilled attacker could use knowledge about the epoch duration used in *ZAPDOS* to craft attacks that inhibit iterative refinement (*e.g.,* by changing attack sources faster than *ZAPDOS's* epoch duration). In addition to keeping actual parameters used in *ZAPDOS* private, production deployments of *ZAPDOS* could also dynamically adjust epoch duration following cryptographically-secure random patterns to mitigate the effectiveness of such attacks.

Model-based attacks. Attackers with detailed knowledge of the data used to train the inference model used in *ZAPDOS* could discover ways to trick the model into making false positive or false negative decisions. In addition to keeping training data used in production private, the models used in *ZAPDOS* could be periodically re-trained based on the most recent features of benign traffic observed on the victim's network to reduce the opportunity for such model-based attacks.

10. Summary

In this work, we developed *ZAPDOS*, a novel approach to detecting volumetric DDoS attack signatures in edge

networks using programmable switch hardware. We demonstrated how *ZAPDOS's* prefix-level approach requires modest hardware resource overheads, can detect 99% of attack traffic in less than 25 s with low false-positives, improves efficiency of DDoS defense compared to state-of-the-art by reducing error rates by two to six orders of magnitude, and defends against modern multi-vector and dynamic attacks. *ZAPDOS* enables effective in-network collaborative defense by allowing edge networks to quickly and efficiently generate accurate attack signatures for reporting to upstream ISP or edge-local mitigation.

Acknowledgment

We thank our anonymous shepherd and reviewers for their constructive feedback. This work is supported by the National Science Foundation through OAC-2126281, SaTC-2132651, CICI-2319944, CNS-2212590, CNS-2003257, and OAC-2126327, a grant from Verizon Innovation LLC., a Ripple faculty fellowship, and a Ripple graduate fellowship. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of NSF, Verizon, or Ripple.

References

[1] A. Toh, A. Vij, and S. Pasha, "Azue DDoS protection—2021 Q3 and Q4 DDoS attack trends," https://tinyurl.com/45uwpjem, accessed: 2022.

[2] L. Rudman and B. Irwin, "Characterization and analysis of ntp amplification based ddos attacks," in *2015 Information Security for South Africa (ISSA)*. IEEE, 2015, pp. 1–5.

[3] C. Rossow, "Amplification hell: Revisiting network protocols for ddos abuse." in *NDSS*, 2014, pp. 1–15.

[4] M. Kührer, T. Hupperich, C. Rossow, and T. Holz, "Hell of a handshake: Abusing TCP for reflective amplification DDoS attacks," in *8th USENIX Workshop on Offensive Technologies*, 2014.

[5] ——, "Exit from hell? reducing the impact of Amplification DDoS attacks," in *23rd USENIX Security Symposium*, 2014, pp. 111–125.

[6] J. Caballero, C. Grier, C. Kreibich, and V. Paxson, "Measuring Pay-per-Install: The commoditization of malware distribution," in *20th USENIX Security Symposium*, 2011.

[7] A. Wang, W. Chang, S. Chen, and A. Mohaisen, "Delving into internet ddos attacks by botnets: characterization and analysis," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2843–2855, 2018.

[8] A. Wang, A. Mohaisen, and S. Chen, "An adversary-centric behavior modeling of ddos attacks," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 1126–1136.

[9] "DDoS attacks: a big problem for small business," https://www.ratcliff.it/news/ddos-attacks-a-big-problem-for-small-business, accessed: 2023-12-06.

[10] "Reactive DDoS defense services," https://www.business.att.com/products/ddos-protection.html, accessed: 2022.

[11] "Cloudflare DDoS protection & mitigation," https://www.cloudflare.com/ddos-hub/, accessed: 2022.

[12] "Blackholing advanced - optimize your response to DDoS attacks," https://www.de-cix.net/en/services/blackholing-advanced, accessed: 2022.

[13] M. Wichtlhuber, E. Strehle, D. Kopp, L. Prepens, S. Stegmueller, A. Rubina, C. Dietzel, and O. Hohlfeld, "Ixp scrubber: learning from blackholing traffic for ml-driven ddos detection at scale," in *Proceedings of the ACM SIGCOMM Conference*, 2022, pp. 707–722.

[14] "DDoS protection solutions - ddos attack mitigation," https://www.netscout.com/solutions/ddos-protection, accessed: 2022.

[15] "DDoS services: Cloud security products and solutions," https://www.radware.com/products/cloud-ddos-services/, accessed: 2022.

[16] "$P4_{16}$ portable switch architecture (PSA): Version 1.1," https://p4.org/p4-spec/docs/PSA-v1.1.0.html, 2018, accessed: 2022.

[17] P. Bosshart, G. Gibb, H.-S. Kim, G. Varghese, N. McKeown, M. Izzard, F. Mujica, and M. Horowitz, "Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN," *ACM SIGCOMM CCR*, vol. 43, no. 4, pp. 99–110, 2013.

[18] "Intel Tofino," https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch.html, accessed: 2022.

[19] "Trident4 / BCM56880 series," https://www.broadcom.com/products/ethernet-connectivity/switching/strataxgs/bcm56880-series, accessed: 2022.

[20] Z. Liu, H. Namkung, G. Nikolaidis, J. Lee, C. Kim, X. Jin, V. Braverman, M. Yu, and V. Sekar, "Jaqen: A high-performance switch-native approach for detecting and mitigating volumetric ddos attacks with programmable switches," in *30th USENIX Security Symposium*, 2021.

[21] A. da Silveira Ilha, Â. C. Lapolli, J. A. Marques, and L. P. Gaspary, "Euclid: A fully in-network, p4-based approach for real-time ddos attack detection and mitigation," *IEEE Transactions on Network and Service Management*, 2020.

[22] A. G. Alcoz, M. Strohmeier, V. Lenders, and L. Vanbever, "Aggregate-based congestion control for pulse-wave ddos defense," in *Proceedings of the ACM SIGCOMM Conference*, 2022, pp. 693–706.

[23] J. Xing, W. Wu, and A. Chen, "Ripple: A programmable, decentralized link-flooding defense against adaptive adversaries," in *30th USENIX Security Symposium*, 2021.

[24] R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martinez-del Rincon, and D. Siracusa, "Lucid: A practical, lightweight deep learning solution for ddos attack detection," *IEEE Transactions on Network and Service Management*, vol. 17, no. 2, pp. 876–889, 2020.

[25] D. Barradas, N. Santos, L. Rodrigues, S. Signorello, F. M. Ramos, and A. Madeira, "Flowlens: Enabling efficient flow classification for ml-based network security applications." in *NDSS*, 2021.

[26] A. T.-J. Akem, M. Gucciardo, M. Fiore *et al.*, "Flowrest: Practical flow-level inference in programmable switches with random forests," in *IEEE International Conference on Computer Communications*, 2023.

[27] G. Xie, Q. Li, Y. Dong, G. Duan, Y. Jiang, and J. Duan, "Mousika: Enable general in-network intelligence in programmable switches by knowledge distillation," in *IEEE International Conference on Computer Communications*, 2022.

[28] E. Kohler, J. Li, V. Paxson, and S. Shenker, "Observed structure of addresses in ip traffic," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, 2002, pp. 253–266.

[29] ——, "Observed structure of addresses in ip traffic," *IEEE/ACM Transactions on Networking*, vol. 14, no. 6, pp. 1207–1218, 2006.

[30] P. Barford, R. Nowak, R. Willett, and V. Yegneswaran, "Toward a model for source addresses of internet background radiation," in *Proc. of the Passive and Active Measurement Conference*, 2006.

[31] A. Gupta, R. Harrison, M. Canini, N. Feamster, J. Rexford, and W. Willinger, "Sonata: Query-driven streaming network telemetry," in *Proceedings of the ACM SIGCOMM Conference*, 2018, pp. 357–371.

[32] M. Karami and D. McCoy, "Understanding the emerging threat of DDoS-as-a-Service," in *6th USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET 13)*, 2013.

[33] M. Karami, Y. Park, and D. McCoy, "Stress testing the booters: Understanding and undermining the business of ddos services," in *Proceedings of the 25th International Conference on World Wide Web*, 2016, pp. 1033–1043.

[34] M. Anagnostopoulos, S. Lagos, and G. Kambourakis, "Large-scale empirical evaluation of dns and ssdp amplification attacks," *Journal of Information Security and Applications*, vol. 66, p. 103168, 2022.

[35] J. J. Santanna, R. van Rijswijk-Deij, R. Hofstede, A. Sperotto, M. Wierbosch, L. Z. Granville, and A. Pras, "Booters—an analysis of ddos-as-a-service attacks," in *2015 IFIP/IEEE International Symposium on Integrated Network Management*. IEEE, 2015, pp. 243–251.

[36] A. Welzel, C. Rossow, and H. Bos, "On measuring the impact of ddos botnets," in *Proceedings of the Seventh European Workshop on System Security*, 2014, pp. 1–6.

[37] J. Zheng, Q. Li, G. Gu, J. Cao, D. K. Yau, and J. Wu, "Realtime ddos defense using cots sdn switches via adaptive correlation analysis," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 7, pp. 1838–1853, 2018.

[38] M. Zhang, G. Li, S. Wang, C. Liu, A. Chen, H. Hu, G. Gu, Q. Li, M. Xu, and J. Wu, "Poseidon: Mitigating volumetric ddos attacks with programmable switches," in *the 27th Network and Distributed System Security Symposium*, 2020.

[39] A. Febro, H. Xiao, and J. Spring, "Distributed sip ddos defense with p4," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2019, pp. 1–8.

[40] X. Z. Khooi, L. Csikor, D. M. Divakaran, and M. S. Kang, "Dida: Distributed in-network defense architecture against amplified reflection ddos attacks," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, 2020, pp. 277–281.

[41] G. Cormode and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications," *Journal of Algorithms*, vol. 55, no. 1, pp. 58–75, 2005.

[42] M. Yu, L. Jose, and R. Miao, "Software defined traffic measurement with OpenSketch," in *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2013, pp. 29–42.

[43] Q. Huang, P. P. Lee, and Y. Bao, "Sketchlearn: Relieving user burdens in approximate measurement with automated statistical inference," in *Proceedings of the ACM SIGCOMM Conference*, 2018, pp. 576–590.

[44] X. Jing, J. Zhao, Q. Zheng, Z. Yan, and W. Pedrycz, "A reversible sketch-based method for detecting and mitigating amplification attacks," *Journal of Network and Computer Applications*, vol. 142, pp. 15–24, 2019.

[45] R. Schweller, Z. Li, Y. Chen, Y. Gao, A. Gupta, Y. Zhang, P. A. Dinda, M.-Y. Kao, and G. Memik, "Reversible sketches: enabling monitoring and analysis over high-speed data streams," *IEEE/ACM Transactions on Networking*, vol. 15, no. 5, pp. 1059–1072, 2007.

[46] M. Asad, M. Asim, T. Javed, M. O. Beg, H. Mujtaba, and S. Abbas, "Deepdetect: detection of distributed denial of service attacks using deep learning," *The Computer Journal*, vol. 63, no. 7, pp. 983–994, 2020.

[47] X. Yuan, C. Li, and X. Li, "Deepdefense: identifying ddos attack via deep learning," in *2017 IEEE international conference on smart computing (SMARTCOMP)*, 2017, pp. 1–8.

[48] T. A. Tuan, H. V. Long, L. H. Son, R. Kumar, I. Priyadarshini, and N. T. K. Son, "Performance evaluation of botnet ddos attack detection using machine learning," *Evolutionary Intelligence*, vol. 13, no. 2, pp. 283–294, 2020.

[49] Y. Feng and J. Li, "Toward explainable and adaptable detection and classification of distributed denial-of-service attacks," in *International Workshop on Deployable Machine Learning for Security Defense*. Springer, 2020, pp. 105–121.

[50] I. L. Meitei, K. J. Singh, and T. De, "Detection of ddos dns amplification attack using classification algorithm," in *Proceedings of the International Conference on Informatics and Analytics*, 2016, pp. 1–6.

[51] G. Zhou, Z. Liu, C. Fu, Q. Li, and K. Xu, "An efficient design of intelligent network data plane," in *32nd USENIX Security Symposium*, 2023.

[52] T. Swamy, A. Rucker, M. Shahbaz, I. Gaur, and K. Olukotun, "Taurus: a data plane architecture for per-packet ml," in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 2022, pp. 1099–1114.

[53] T. M. Gil and M. Poletto, "MULTOPS: A data-structure for bandwidth attack detection," in *USENIX Security Symposium*, 2001.

[54] F. Soldo, A. Markopoulou, and K. Argyraki, "Optimal filtering of source address prefixes: Models and algorithms," in *IEEE INFOCOM*, 2009, pp. 2446–2454.

[55] G. Pack, J. Yoon, E. Collins, and C. Estan, "On filtering of ddos attacks based on source address prefixes," in *2006 Securecomm and Workshops*. IEEE, 2006, pp. 1–12.

[56] C. Estan, S. Savage, and G. Varghese, "Automatically inferring patterns of resource consumption in network traffic," in *ACM SIGCOMM*, 2003.

[57] L. Yuan, C.-N. Chuah, and P. Mohapatra, "ProgME: towards programmable network measurement," *IEEE/ACM Transactions on Networking*, vol. 19, no. 1, pp. 115–128, 2011.

[58] M. Moshref, M. Yu, R. Govindan, and A. Vahdat, "DREAM: Dynamic resource allocation for software-defined measurement," *ACM SIGCOMM CCR*, vol. 44, no. 4, pp. 419–430, 2014.

[59] ——, "Scream: Sketch resource allocation for software-defined measurement," in *ACM CoNEXT*, 2015.

[60] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Controlling high bandwidth aggregates in the network," *ACM SIGCOMM CCR*, vol. 32, no. 3, pp. 62–73, 2002.

[61] L. Jose, M. Yu, and J. Rexford, "Online measurement of large traffic aggregates on commodity switches," in *Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services*, Mar. 2011.

[62] M. S. Kang, V. D. Gligor, V. Sekar *et al.*, "Spiffy: Inducing cost-detectability tradeoffs for persistent link-flooding attacks." in *NDSS*, vol. 1, 2016, pp. 53–55.

[63] H. Zhou, S. Hong, Y. Liu, X. Luo, W. Li, and G. Gu, "Mew: Enabling large-scale and dynamic link-flooding defenses on programmable switches," in *IEEE Symposium on Security and Privacy*, 2023, pp. 3178–3192.

[64] M. S. Kang, S. B. Lee, and V. D. Gligor, "The crossfire attack," in *IEEE symposium on security and privacy*, 2013, pp. 127–141.

[65] A. Studer and A. Perrig, "The coremelt attack," in *European Symposium on Research in Computer Security*. Springer, 2009, pp. 37–52.

[66] B. Stone-Gross, C. Kruegel, K. Almeroth, A. Moser, and E. Kirda, "Fire: Finding rogue networks," in *2009 Annual Computer Security Applications Conference*. IEEE, 2009, pp. 231–240.

[67] "DROP - don't route or peer lists - the spamhaus project," https://www.spamhaus.org/drop/, accessed: 2022.

[68] "FRGP (www.frgp.net) continuous flow dataset, IMPACT ID: USC-LANDER/Mirai-FRGP-scanning-20160908/rev10326," provided by the USC/LANDER project (http://www.isi.edu/ant/lander), traces taken 2016-09-08 to 2016-10-31.

[69] R. Fontugne, P. Borgnat, P. Abry, and K. Fukuda, "MAWILab: Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking," in *Proceedings of the ACM Conference on emerging Networking EXperiments and Technologies*, 2010.

[70] D. Wagner, D. Kopp, M. Wichtlhuber, C. Dietzel, O. Hohlfeld, G. Smaragdakis, and A. Feldmann, "United we stand: Collaborative detection and mitigation of amplification ddos attacks at scale," in *Proceedings of the 2021 ACM SIGSAC conference on computer and communications security*, 2021, pp. 970–987.

[71] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *IEEE symposium on security and privacy*, 2010, pp. 305–316.

[72] Y. Lavinia, R. Durairajan, R. Rejaie, and W. Willinger, "Challenges in using ml for networking research: How to label if you must," in *Proceedings of the Workshop on Network Meets AI & ML*, 2020, pp. 21–27.

[73] "The CAIDA UCSD "DDoS attack 2007" dataset," http://www.caida.org/data/passive/ddos-20070804_dataset.xml, accessed: 2022.

[74] Z. Xu, S. Ramanathan, A. Rush, J. Mirkovic, and M. Yu, "Xatu: boosting existing ddos detection systems using auxiliary signals," in *Proceedings of the 18th International Conference on emerging Networking EXperiments and Technologies*, 2022, pp. 1–17.

[75] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." *ICISSp*, vol. 1, pp. 108–116, 2018.

[76] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers & Security*, vol. 31, no. 3, pp. 357–374, 2012.

[77] "IDS 2018," https://www.unb.ca/cic/datasets/ids-2018.html, accessed: 2023-06-22.

[78] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, "Developing realistic distributed denial of service (ddos) attack dataset and taxonomy," in *2019 International Carnahan Conference on Security Technology (ICCST)*. IEEE, 2019, pp. 1–8.

[79] R. Montoro, "Quick analysis of a DDoS attack using ssdp," https://blog.sucuri.net/2014/09/quick-analysis-of-a-ddos-attack-using-ssdp.html, 2022.

[80] T. A. Tuan, H. V. Long, R. Kumar, I. Priyadarshini, N. T. K. Son *et al.*, "Performance evaluation of botnet ddos attack detection using machine learning," *Evolutionary Intelligence*, pp. 1–12, 2019.

[81] D. Stiawan, M. Y. B. Idris, A. M. Bamhdi, R. Budiarto *et al.*, "Cicids-2017 dataset feature analysis with information gain for anomaly detection," *IEEE Access*, vol. 8, pp. 132 911–132 921, 2020.

[82] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[83] A. Cutler, D. R. Cutler, and J. R. Stevens, "Random forests," in *Ensemble machine learning*. Springer, 2012, pp. 157–175.

[84] "randomForest: Breiman and cutler's random forests for classification and regression," https://cran.r-project.org/web/packages/randomForest/randomForest.pdf, accessed: 2022.

[85] Y. Zhou, D. Zhang, K. Gao, C. Sun, J. Cao, Y. Wang, M. Xu, and J. Wu, "Newton: Intent-driven network traffic monitoring," in *Proceedings of the ACM Conference on emerging Networking EXperiments and Technologies (CoNEXT)*, 2020, pp. 295–308.

[86] C. Misa, W. O'Connor, R. Durairajan, R. Rejaie, and W. Walter, "Dynamic scheduling of approximate telemetry queries," in *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation*, 2022, pp. 701–717.

[87] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[88] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM CCR*, vol. 44, no. 3, pp. 87–95, 2014.

[89] "Haskell language," https://www.haskell.org/, accessed 2023-12-05.

[90] "In-band network telemetry (int) dataplane specification," https://p4.org/p4-spec/docs/INT_v2_1.pdf, accessed 2023-12-05.

[91] D. Bernier, "Rfc 9378: In situ operations, administration, and maintenance (ioam) deployment," 2023.

[92] C. Shou, R. Bhatia, A. Gupta, R. Harrison, D. Lokshtanov, and W. Willinger, "Query planning for robust and scalable hybrid network telemetry systems," *Proceedings of the ACM on Networking*, vol. 2, pp. 1–27, 2024.

[93] L. Ablon, M. C. Libicki, and A. A. Golay, *Markets for cybercrime tools and stolen data: Hackers' bazaar*. Rand Corporation, 2014.

[94] Z. Li and Q. Liao, "Toward a monopoly botnet market," *Information Security Journal: A Global Perspective*, vol. 23, no. 4-6, pp. 159–171, 2014.

[95] "Dark web price index 2022," https://www.privacyaffairs.com/dark-web-price-index-2022/, accessed: 2022.

[96] V. Segura and J. Lahuerta, "Modeling the economic incentives of ddos attacks: Femtocell case study," in *Economics of information security and privacy*. Springer, 2010, pp. 107–119.

Appendix A. Prefix-Level Attack Signatures

The methods developed in *ZAPDOS* are based on observations that both attack and benign sources are not uniformly distributed in the source address space, but tend to cluster in distinctive prefix-level structures formally described by multifractal models [28]–[30]. While preliminary evidence of multifractal clustering of benign Internet traffic has been reported in [28], [29], in this section we provide concrete arguments for why certain types of volumetric DDoS attack traffic exhibit a similar clustering and the implications for attack signature detection. In particular, we consider the two most common volumetric DDoS attack scenarios: reflection-based attacks (§ A.1) and botnet-based attacks (§ A.2).

A.1. Reflection-Based Attacks

The first common approach to generating large amounts of traffic for a volumetric DDoS attack uses a set of publicly accessible servers, known as reflectors or amplifiers, to reflect and amplify attack traffic towards a victim [3]. In particular, the attacker sends requests to these servers with the source address of the request set to the address of the victim so that the server's response, which is typically much larger than the attacker's request, is forwarded to the victim. Note that in reflection attacks, the victim sees traffic coming the addresses of the servers used as reflectors and does not directly observe the addresses responsible for launching the attack.

Clustering of effective reflectors. Our first observation is that, while attackers can freely choose which reflectors to use, they cannot choose the IP addresses of particular reflectors. Intuitively, since effective reflectors are typically misconfigured servers with high-bandwidth connections and stable up-time [33], reflectors can be expected to cluster in address regions associated with networks that have a particular type of policy (e.g., lax policies about updates/patches coupled with high-bandwidth connections to the rest of the Internet). For example, Figure A.1 shows a scenario where an attacker has identified DNS servers to use for reflection

in 10.30.0.0/16 and 10.20.0.0/16 (*i.e.,* the "−" partition of the address space where it is easy to launch attacks from). However, a majority of the defender's clients are in 10.10.0.0/16 (*i.e.,* the "+" partition of the address space) with only one client co-located in a prefix abused by the attacker (10.20.0.0/16). Clearly an optimal defense in this scenario would seek to focus resources on separating the mix of benign and attack traffic from 10.20.0.0/16 rather than treating each prefix uniformly.

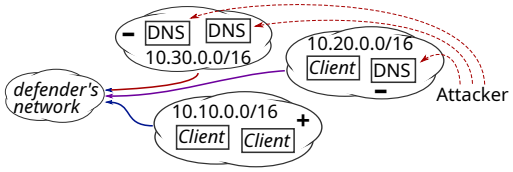


Figure A.1: Example of "+" and "−" partitions of the source address space in a DNS reflection attack.

The existence of networks whose policies support abusable services (like the DNS servers of 10.20.0.0/16 and 10.30.0.0/16 in Figure A.1) is unfortunately widely accepted [66], [67] and several works analyzing actual DDoS attacks purchased from booters services [35], as well as leaked operation data bases of booters [32], [33] confirm that reflectors actually used by such services do tend to cluster in a few specific ASes. For example, [33] found that a single AS was responsible for ∼20% of all SSDP amplifiers used by a particular booter service.

Relationship to benign traffic. Because most publicly-available datasets contain only anonymized IP addresses, it is hard to estimate the degree to which the clustering of benign traffic might overlap with the clustering of effective reflectors used for DDoS attacks (*i.e.,* the prevalence of prefixes like 10.20.0.0/16 from Figure A.1). However, we argue that it is unlikely for the types of networks that host large numbers of effective collectors to overlap significantly with networks that host the benign clients of enterprise or campus networks.

A.2. Botnet-Based Attacks

The second common approach for launching volumetric DDoS attacks is to leverage a large set of infected hosts known as a botnet to directly send a flood of traffic towards the victim [7], [36]. Since bots are infected hosts in otherwise non-malicious networks, we assume they do not spoof their source addresses, an assumption confirmed by several empirical studies of botnets used for launching DDoS attacks [7], [8]. As a result, we assume the victim sees traffic coming directly from bot addresses.

Different costs for different regions. The simplest and most common means for an attacker to gain access to a sufficiently large botnet is to simply purchase access through a pay-per-install (PPI) market place [6], [93]. However, due to strong free-market competition [93], [94], prices for bots in different regions of the IP address space can vary drastically. For example, Figure A.2 shows a scenario where the cost for the attacker to access a bot in 10.50.0.0/16 or 10.40.0.0/16 (*i.e.,* the "+" partition of the address space) is far greater than the cost to access a bit in 10.60.0.0/16 (*i.e.,* the "−" partition of the address space). An optimal defense in this scenario would focus resources on separating attack and benign traffic coming from 10.40.0.0/16 and 10.50.0.0/16 and would simply block 10.60.0.0/16 with a single TCAM entry.

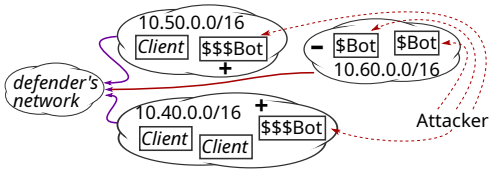


Figure A.2: Example of "+" and "−" partitions of the source address space in a botnet-based attack.

Differing prices based on region are commonly used to distinguish between service tiers in the PPI market place. For example, [95][12] and [6] both report a roughly order of magnitude difference between location-agnostic installs and installs in US-based regions. Moreover, there is strong reason to believe that such price differentiation is a natural and persistent result of the economic model of the PPI market place [94], [96] (intuitively, strong competition between different PPI providers fosters service differentiation and pricing by region is a commonly implement form of such differentiation).

Appendix B. Generic Reflection Attack Features

In this section we describe the generic $rrDiff$ feature which summarises multiple well-known signatures at the prefix-level. Intuitively, $rrDiff$ captures patterns where attack traffic tends to have a larger number of packets of one class (*e.g.,* responses) compared to another class (*e.g.,* requests). (Note that we will refer to these two classes as "responses" and "requests" in the following.)

In particular, for each prefix p and each attack vector v, we compute the total number of request packets ($req_{p,v}$) and the total number of response packets ($resp_{p,v}$), respectively. Table B.1 shows the set of response and request entries for the attack vectors considered in this work along with details about how we classify packets as being requests or responses for each vector. Given these sums that the switch hardware computes for a given prefix p, we consider the feature $rrDiff_p$ that summarizes the degree of response-request imbalance in that prefix and is defined as

$$rrDiff_p = \max_v \left(resp_{p,v} - req_{p,v}\right)$$

Intuitively, the value of $rrDiff_p$ will be high if p sources attack traffic and close to zero if p sources only benign traffic.

12. Though absolute prices have increased, the difference between regions over the last three years of the dark web price index [95] have remained relatively constant.

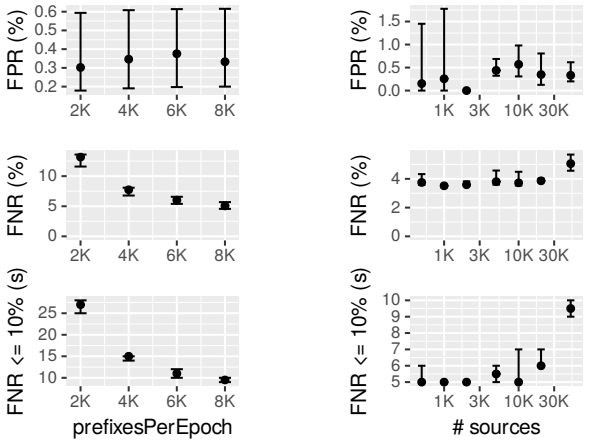
13. Note that SYN floods are not technically a reflection attack, but we can still capture their asymmetry with our response/request difference feature.

Protocol	Request	Response
DNS refl.	UDP to port 53	UDP from port 53
NTP refl.	UDP to port 123	UDP from port 123
SSDP refl.	UDP to port 1900	UDP from port 1900
SYN flood[13]	TCP, SYN+ACK set	TCP, only SYN set

TABLE B.1: List of currently considered reflection vectors.

Appendix C. Extended Evaluation

Since *ZAPDOS* uses a limited amount of switch hardware memory to monitor attack prefixes, we evaluate how changes to the amount of memory used and to the number of distinct attack sources impact performance. We repeat these evaluations over all 6 attack vectors from Table 3 and show the minimum, median, and maximum values in Figure C.3.



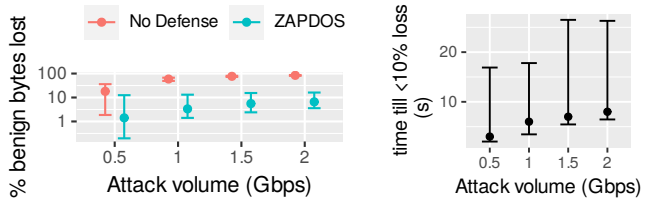
(a) Varying resources. (b) Varying attack parameter.

Figure C.3: Performance of *ZAPDOS* while varying `prefixesPerEpoch` (left) and the number of distinct attack sources (right).

Impact of resource constraints. In Figure C.3a, we vary `prefixesPerEpoch` while keeping the number of attack sources constant at 50 k. In all cases, *ZAPDOS* achieves low FPR (0.3% to 0.4%) independent of `prefixesPerEpoch`. On the other hand, FNR as well as time until FNR is less than 10% decreases as `prefixesPerEpoch` increases since monitoring more prefixes each epoch enables faster progress towards identifying all attack sources.

Impact of attack sources. In Figure C.3b, we vary the number of distinct sources in attack traffic while keeping `prefixesPerEpoch` at 8 k. *ZAPDOS* achieves consistently low error rates ($\sim$ 0.5% FPR and $\sim$3.5% FNR) for a wide range of number of sources (500 through 10 k). However, as the number of sources increases, both total FNR and the time until per-epoch FNR drops below 10% begin increasing. We recall that the total attack traffic rate is held constant across these scenarios so that with more sources, the per-source attack rate is significantly reduced. As reflected in Figure C.3b, this decrease in per-source attack rate makes the detection problem harder by reducing the relative strength of the attack signals compared to benign traffic hence increasing *ZAPDOS's* latency and FNR.

Handling Flooded Border Links A volumetric DDoS attack against a small enterprise or campus network at the edge of the Internet can flood the network's main border link, rendering in-network mitigation methods (*e.g.,* Jaqen, Euclid) ineffective and degrading the accuracy of traffic monitoring performed at the edge network. To demonstrate the utility of *ZAPDOS's* attack signatures in this scenario, we simulate a 1 Gbps border link inserted in front of where *ZAPDOS* performs its traffic monitoring computations (using a simulated 100 KB FIFO queue). We generate DNS reflection attacks using random samples of 10k sources[14] from the Booters [35] dataset and varying the per-source attack data rate so that the overall attack rate varies from 500 to 2000 Mbps. We simulate upstream mitigation by dropping all packets matching *ZAPDOS's* reported attack signature.



(a) Total bytes of benign traffic lost with and without *ZAPDOS*. (b) Time till benign bytes lost drops below 10%.

Figure C.4: Impact of DNS refl. attack volume on defense performance given a 1 Gbps upstream bottleneck link. *ZAPDOS* reduces benign traffic loss to less than 10% in less than 10 s for the strongest attack (2 Gbps).

Figure C.4 shows the performance of *ZAPDOS* over 10 independent attacks at each attack volume in terms of total damage (volume of benign traffic dropped by the flooded link) and the time it takes before *ZAPDOS* reduces per-epoch damage to below 10%. We observe that with no defense, the volume of dropped benign traffic increases consistently with increasing attack volume. On the other hand, Figure C.4a shows that signature-based upstream mitigation enabled by *ZAPDOS* reduces flooding-induced loss to below 10% (median over 10 attacks), even for the largest attack volume (which is 2$\times$ the border link's throughput and incurs $\sim$85% loss without defense). For all attack volumes, *ZAPDOS* reduces benign loss more than 10$\times$.

Figure C.4b shows that larger attacks require longer refinement time before the attack signature is specific enough to reduce loss to below 10%. We observe a similar trend for other percentages of benign traffic loss. This is due to the fact that even though *ZAPDOS* refines signatures at a similar rate for all attack volumes, in larger attacks, individual attack sources send larger volumes of attack traffic so that a larger number of sources must be blocked before the overall reduction in attack traffic (and hence damage to benign traffic) reaches the same level as for smaller attacks. Nonetheless, even for the 2 Gbps attack, *ZAPDOS* reduces loss to below 10% in less than 10 s on average.

14. Note that we use a smaller number of sources compared to, *e.g.,* Figure 12 due to the smaller number of sources available in the Booters dataset.

Appendix D.
Meta-Review

The following meta-review was prepared by the program committee for the 2024 IEEE Symposium on Security and Privacy (S&P) as part of the review process as detailed in the call for papers.

D.1. Summary

This paper presents ZAPDOS, a system that targets detection of DDoS attack signatures with a unique dual hardware-software approach. For scalable and efficient detection, ZAPDOS monitors source prefixes in high-speed programmable switches, and uses signature-based ML classification with other heuristics to identify and pinpoint suspicious prefixes.

D.2. Scientific Contributions

- Addresses a Long-Known Issue
- Provides a Valuable Step Forward in an Established Field

D.3. Reasons for Acceptance

1) This paper addresses a long-known issue. Defending against DDoS attacks remains a challenge today, and this paper offers a novel defense paradigm targeted for small to mid-sized enterprises with reduced resource capacity, which is important and practical.
2) This paper provides a valuable step forward in an established field. It presents a fresh perspective on DDoS detection by considering non-uniform traffic distributions, and validates the practicality of the defense on a programmable switch.