UNDERSTANDING AND ADVANCING THE STATUS QUO OF DDOS DEFENSE

by

LUMIN SHI

A DISSERTATION

Presented to the Computer and Information Science
and the Division of Graduate Studies of the University of Oregon
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

March 2022

DISSERTATION APPROVAL PAGE

Student: Lumin Shi

Title: Understanding and Advancing the Status Quo of DDoS Defense

This dissertation has been accepted and approved in partial fulfillment of the requirements for the Doctor of Philosophy degree in the Computer and Information Science by:

Jun Li	Chair
Reza Rejaie	Core Member
Lei Jiao	Core Member
Bryce Newell	Institutional Representative

and

Krista Chronsiter	Vice Provost for Graduate Studies

Original approval signatures are on file with the University of Oregon Division of Graduate Studies.

Degree awarded March 2022

DISSERTATION ABSTRACT

Lumin Shi

Doctor of Philosophy

Computer and Information Science

March 2022

Title: Understanding and Advancing the Status Quo of DDoS Defense

Two decades after the first distributed denial-of-service (DDoS) attack, the Internet remains challenged by DDoS attacks as they evolve. Not only is the scale of attacks larger than ever, but they are also harder to detect and mitigate. Nevertheless, the Internet's fundamental design, based on which machines are free to send traffic to any other machines, remains the same. This thesis reinvestigates the prior DDoS defense solutions to find less studied but critical issues in existing defense solutions. It proposes solutions to improve the *input*, *design*, and *evaluation* of DDoS defense. Specifically, we show why DDoS defense systems need a better view of the Internet's traffic at the autonomous system (AS) level. We use a novel attack to expose the inefficiencies in the existing defense systems. Finally, we reason why a defense solution needs a sound empirical evaluation and provide a framework that mimics real-world networks to facilitate DDoS defense evaluation.

This dissertation includes published and unpublished co-authored materials.

CURRICULUM VITAE

NAME OF AUTHOR: Lumin Shi

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

University of Oregon, Eugene, OR, USA
The University of Alabama, Tuscaloosa, AL, USA

DEGREES AWARDED:

Doctor of Philosophy, Computer and Information Science, 2022, University of Oregon
Bachelor of Science, Computer Science, 2014, The University of Alabama

AREAS OF SPECIAL INTEREST:

Network Security
Network Systems
Network Traffic Analysis

PROFESSIONAL EXPERIENCE:

Research and Teaching Assistant, University of Oregon, 2014 - 2021
Information Security Analyst, University of Oregon, 2017 - 2021
Visiting Scholar, Center for Applied Internet Data Analysis (CAIDA), 2019
Web Developer, The University of Alabama, 2012 - 2014

GRANTS, AWARDS AND HONORS:

Phillip Seeley Graduate Fellowship, 2021
Annual Computer Security Applications Conference Travel Grant, 2019
GENI Experimenter Contest Honorable Mention, 2018
Oregon Cyber Security Day Outstanding Demo Price (Co-Winner), 2017

PUBLICATIONS:

Jun Li, Devkishen Sisodia, Yebo Feng, Lumin Shi, Mingwei Zhang, Samuel Mergendahl, Christopher Early, Peter Reiher (2022). Toward Adaptive Distributed Filtering of DDoS Traffic. **In Submission**.

Lumin Shi, Jun Li., Reza Rejaie (2022). WebTell: Identifying Browsed Websites from Network Flow Records. **In Submission**.

Lumin Shi, Devkishen Sisodia, Jun Li, Mingwei Zhang, Alberto Dainotti, Peter Reiher (2022). The Moving Target Attack: On Quantifying Multi-Wave Victims in DDoS Mitigation. **In Preparation**.

Lumin Shi, Jun Li (2022). Two Decades of DDoS Attacks and Defenses. **In Preparation**.

Lumin Shi, Jun Li, Mingwei Zhang, Peter Reiher (2021). On Capturing DDoS Traffic Footprints on the Internet. *IEEE Transactions on Dependable and Secure Computing (TDSC).*

Lumin Shi, Samuel Mergendahl, Devkishen Sisodia, Jun Li (2020). Bridging Missing Gaps in Evaluating DDoS Research. *USENIX Workshop on Cyber Security Experimentation and Test (CSET).*

Lumin Shi, Samuel Mergendahl, Devkishen Sisodia, Jun Li (2020). Playing in the Sandbox: A Step Towards Sound DDoS Research Through High-Fidelity Evaluation (Extended Abstract). *Passive and Active Measurement Conference (PAM).*

Lumin Shi, Devkishen Sisodia, Mingwei Zhang, Jun Li, Alberto Dainotti, Peter Reiher (2019). The Catch-22 Attack (Extended Abstract). *Annual Computer Security Applications Conference (ACSAC).*

Lumin Shi, Mingwei Zhang, Jun Li, Peter Reiher (2019). GENI Experiment Configuration Automated (Technical Report CCSP-TR-2019-02). *Computer and Information Science, University of Oregon.*

Mingwei Zhang, Lumin Shi, Devkishen Sisodia, Jun Li, Peter Reiher (2019). On Multi-Point, In-Network Filtering of Distributed Denial-of-Service Traffic. *IFIP/IEEE International Symposium on Integrated Network Management (IM).*

Lumin Shi, Mingwei Zhang, Jun Li, Peter Reiher (2018). PathFinder: Capturing DDoS Traffic Footprints on the Internet. *IFIP Networking.*

Derek Overlock, Lumin Shi, Xiaoyan Hong, Meng Kuai (2015). Underwater Sensor Network Viewer (Demo Paper). *International Conference on Underwater Networks and Systems*.

ACKNOWLEDGEMENTS

I am grateful to my advisor Prof. Jun Li who guided me to conduct research and taught me lessons that can benefit me in the future. Above all the things I learned from him, he showed me the ways to persevere.

I would also like to thank my other committee members, Prof. Reza Rejaie, Prof. Lei Jiao, and Prof. Bryce Newell, for their valuable comments and suggestions to turn this thesis into a better shape.

I thank all my lab mates at the network and security research group for their countless hours of incredible, intellectual debates in research, as well as their support in everyday life; they are an interesting bunch that helped me to touch the finish line.

Last but not least, I thank my friends at UO Information Services and Center for Applied Internet Data Analysis (CAIDA) for extending me the opportunities to work on real-world network projects; they influenced me on what I want to work on in the future.

To my parents and my wife

TABLE OF CONTENTS

Chapter Page

LIST OF FIGURES

Figure Page

Figure Page

LIST OF TABLES

CHAPTER I

INTRODUCTION

Two decades after the first distributed denial-of-service (DDoS) attack [48], networks on the Internet continue to struggle with DDoS attacks. As the Internet grows in size, the number of end devices vulnerable to attacks also grows significantly. However, by and large, the Internet design remains the same — any machine is free to send traffic to any other machine. As a result, adversaries can launch bigger and more frequent DDoS attacks against various cyber resources. In the time of a global pandemic, more people choose to work from home than ever before. They (and their industries) rely on stable network resources to function. This reliance motivates adversaries to launch more DDoS attacks to meet different goals (e.g., DDoS extortion). Indeed, numerous DDoS mitigation providers report a sharp increase in the number of DDoS attacks in the past year [219, 140, 98]. DDoS defense is more critical than ever. Worse, researchers continue to discover advanced DDoS attacks, such as those described in [86, 88, 175, 179], which are more challenging to defend against with existing defense solutions. It is only a matter of time before more adversaries employ these advanced attacks. In fact, we are starting to see recent DDoS attacks (e.g., carpet bombing attacks [30]) employ basic techniques from the advanced attacks.

Past DDoS attacks led network and research communities to explore effective defense solutions. While DDoS victims (e.g., campus networks) may defend themselves against many attacks locally, they often rely on remote resources to fight off large-scale attacks. In the early days, the network community relied on the remotely triggered black hole (RTBH) [202, 97] to prevent *all* traffic from reaching network prefixes under large-scale attacks. While RTBH protects network prefixes that are not under attack, it blocks all traffic towards the attacked network prefixes, which often completes the attacks for the attackers. More recently, some networks in the community started to utilize solutions such as BGP FlowSpec [123] to filter traffic at a finer granularity. Unfortunately, with only these mitigation solutions, the DDoS defense is blind to the most efficient defense locations on the Internet. Networks may also pay premiums to DDoS protection service (DPS) providers (e.g., Akamai, Cloudflare, Imperva) to cleanse their network traffic at multiple data centers across different countries. While the DPS providers can implement any defense solution due to the extensive reach of their networks on the Internet, their actual defense efficacy is not publicly known. Over the years, the research community produced hundreds of DDoS prevention and mitigation solutions. However, we have yet to see the adoption of these solutions as most of them require a significant number of autonomous systems (ASes) to participate to become effective. Reasons such as inter-AS collaboration, host software modification, and infrequent attack encounters are all plausible reasons that prevent them from being deployed. Because of the lack of deployment, the research community can hardly measure their empirical defense efficacy. For the above reasons, researchers later begin to design defense solutions [53, 109, 62, 44] that exploit the Internet routing protocol (i.e., Border Gateway Protocol (BGP)) to reduce their deployment friction on the Internet. Still, their technical underpinning remains an active research area [92, 228]. In other words, the research community still needs to improve the defense efficacy with the above easy-to-deploy defense solutions.

1.1 Dissertation Scope & Thesis Statement

What makes DDoS attacks so challenging to defend against that these attacks continue to plague the Internet after two decades of research? We approach this question by first understanding the state-of-the-art DDoS defense solutions, then study what an effective DDoS defense solution should achieve from three complementary aspects: **input**, **design**, and **evaluation**. Specifically, we study (1) if and how an effective DDoS defense can be facilitated with better input, (2) if and how current DDoS defense solutions may actually be exploited by adversaries in real settings, and (3) if and how the evaluation of DDoS defense solutions can be improved with a higher fidelity. With the help of public datasets online, we run many data-driven simulation and emulation experiments to quantify each study above.

The thesis statement is as follows. **The state of the art of DDoS defense needs to be strengthened with (1) more timely and informative DDoS traffic footprint information, (2) stronger resilience against sophisticated attackers in real-world defense, and (3) sound evaluation environment with a higher fidelity that facilitates empirical analysis.**

We elaborate each aspect below.

1.2 On Capturing DDoS Traffic Footprints on the Internet – Toward Better DDoS Defense Input

While distributed denial-of-service (DDoS) attacks are easy to launch and are becoming more damaging, the defense against DDoS attacks often suffers from the lack of relevant knowledge of the DDoS traffic, including the paths the DDoS traffic has traversed, the source addresses (spoofed or not) that appear along each path, and the amount of traffic per path or per source. Though IP traceback and path inference approaches could be considered, they are either expensive and hard to deploy or inaccurate. We propose PathFinder, a service that a DDoS defense system can use to obtain the footprints of the DDoS traffic to the victim. PathFinder employs an architecture that is easy to implement and deploy on today's Internet, a PFTrie data structure that introduces multiple design features to log traffic footprints at line rate, and streaming and zooming mechanisms that facilitate the storage and transmission of DDoS footprints more efficiently. Our evaluation shows that PathFinder can significantly improve the efficacy of a DDoS defense system by placing traffic filters at the right defense locations, its PFTrie data structure is fast and has a manageable overhead, and its streaming and zooming mechanisms significantly reduce the delay and overhead in transmitting DDoS footprints.

1.3 Moving Target Attack Design and Analysis – Toward Understanding the Inadequacies of DDoS Defense Design in Real World

In this work, we examine how an attacker, with a finite amount of resources, can launch a link-flooding attack to impose a mitigation dilemma on multiple attacked ASes simultaneously and expose the inadequacies of DDoS mitigation solutions. We name this attack the moving target attack (MTA). In MTA, each attacked AS either endures the attack or bear undesirable collateral damage in attack mitigation. Consisting of components that are practical to implement, MTA strives to maximize the amount of strain on DDoS defense systems and the collateral damage incurred by defending networks, thereby wreaking havoc on wide swaths of the Internet. We survey DDoS mitigation solutions in the real world, examine their operational requirements, capabilities, and implications. We then evaluate MTA against the mitigation solutions from our survey. By leveraging real-world datasets, we illustrate the feasibility of the attack and quantify the amount of damage it can impose on today's Internet. We thus hope that this work can motivate the network community to address the design issues of existing DDoS defense systems.

1.4 DDoS SandBox – Toward High-Fidelity DDoS Defense Evaluation

While distributed denial-of-service (DDoS) attacks become stealthier and more disruptive, real-world network operators often ignore academic DDoS defense research and instead rely on basic defense techniques that cannot adequately defend them in many situations. In fact, prior to the deployment of a DDoS defense solution, a network operator must understand its impact specifically on their network. However, with only carefully prepared experimentation settings in closed environment but without a sound empirical analysis of the solution, which is often the case even for the most cited academic research, the network operator may fear its defense efficacy is poor in real world due to known issues such as an increased false-positive rate from domain shift or its adverse effects on legitimate traffic from coarse-grained mitigation techniques.

In this work, we elaborate on the critical missing gaps in DDoS defense evaluation and propose an emulation-based evaluation platform to help produce the missing defense analytics. Specifically, the platform is to identify the impact of a defense solution in realistic network settings. It offers to emulate a mini Internet topology with realistic IP address space allocation and generate representational background traffic specific to particular networks. Last but not least, because it is an emulation platform with the above properties, an experimenter can run virtually any applications as if they are deployed on the real Internet; such a feature is especially important for network operators to understand if an academic solution is mature enough through empirical studies. As such, our platform fulfills the prominent missing gaps in current DDoS research.

1.5 Dissertation Outline

The dissertation is organized as follows. We first provide a background to DDoS and survey the DDoS attacks and defense strategies published in the past two decades in Chapter II and Chapter III, respectively. Next, in Chapter IV we propose a system for DDoS defense systems to collect traffic footprints on behalf of a DDoS victim, and quantify the importance of a good input to DDoS defense systems. In Chapter V, we present a practical and novel DDoS attack that exposes DDoS mitigation issues in the real world quantitatively, hoping the evaluation results can sound an alarm to the network community. We then elaborate on the critical missing gaps in the DDoS defense evaluation and propose a new evaluation platform (i.e., the DDoS SandBox) in Chapter VI. The platform allows its users (i.e., network operators, researchers) to evaluate DDoS attacks and defense solutions in a closed-loop environment. Finally, we discuss the future work that can further the DDoS research in Chapter VII and conclude our contributions in Chapter VIII.

1.6 Co-authored Materials & Acknowledgment

1.6.1 Co-authored Materials. Most content in this thesis is from our published and unpublished work. Below we connect each related chapter to the material and authors that contributed.

- Chapter II and Chapter III:
  * Unpublished as Lumin Shi and Jun Li. Two Decades of DDoS Attacks and Defenses.

- Chapter IV:
  - Published as Lumin Shi, Jun Li, Mingwei Zhang, Peter Reiher. On Capturing DDoS Traffic Footprints on the Internet. *IEEE Transactions on Dependable and Secure Computing*, 2021.
  - Published as Lumin Shi, Mingwei Zhang, Jun Li, Peter Reiher. PathFinder: Capturing DDoS Traffic Footprints on the Internet. *IFIP Networking*, 2018.
- Chapter V:
  - Published as Lumin Shi, Devkishen Sisodia, Mingwei Zhang, Jun Li, Alberto Dainotti, Peter Reiher. The Catch-22 Attack (Extended Abstract). *Annual Computer Security Applications Conference*, 2019.
  - Unpublished as Lumin Shi, Devkishen Sisodia, Jun Li, Mingwei Zhang, Alberto Dainotti, Peter Reiher. The Moving Target Attack: On Quantifying Multi-Wave Victims in DDoS Mitigation.
- Chapter VI:
  - Published as Lumin Shi, Samuel Mergendahl, Devkishen Sisodia, Jun Li. Bridging Missing Gaps in Evaluating DDoS Research. *13th USENIX Workshop on Cyber Security Experimentation and Test*, 2020.
  - Published as Lumin Shi, Samuel Mergendahl, Devkishen Sisodia, Jun Li. Playing in the Sandbox: A Step Towards Sound DDoS Research Through High-Fidelity Evaluation (Extended Abstract). *Passive and Active Measurement Conference*, 2020.

1.6.2 Acknowledgment. This project is in part the result of funding provided by the Science and Technology Directorate of the United States Department of Homeland Security under contract number D15PC00204. The views and conclusions contained herein are those of the authors and should not be interpreted necessarily representing the official policies or endorsements, either expressed or implied, of the Department of Homeland Security or the US Government.

CHAPTER II

BACKGROUND

The chapter begins with an introduction to DDoS attacks and defense solutions, as well as their taxonomies in Sec. 2.1. The introduction illustrates the complexity in DDoS defense. We then dive into the advanced attacks and generalize their attack models in Sec. 2.2. The section assesses each attack's goal, assumptions, methodology, and practicality.

The content in this chapter appeared in the following unpublished work: Shi, L.; Li, J., Two Decades of DDoS Attacks and Defenses. Lumin Shi is the leading author of this work and responsible for leading all the presented analyses.

2.1 DDoS Primer

This section introduces DDoS attacks, defense strategies, and their complexities. We start with a simplified Internet topology to demonstrate the critical resources that an attacker can target (Sec. 2.1.1). Then, we cover the main factors the attacker may consider before launching an attack (Sec. 2.1.2). We conclude this section with a DDoS attack taxonomy (Sec. 2.1.3) and a DDoS defense taxonomy (Sec. 2.1.4). The former clarifies the relationships of different DDoS attacks. For example, what is the relationship between a volumetric attack and a domain name system (DNS) amplification attack? The latter classifies DDoS defense solutions into three defense strategies and cover their advantages and disadvantages.

2.1.1 Attack Targets. The goal of a DDoS attack is to render a service (e.g., a video streaming platform) inaccessible to its benign users. To deliver a service on the Internet, one or more hosts (i.e., servers) need to process user requests and return the corresponding information back to users. Therefore, if an

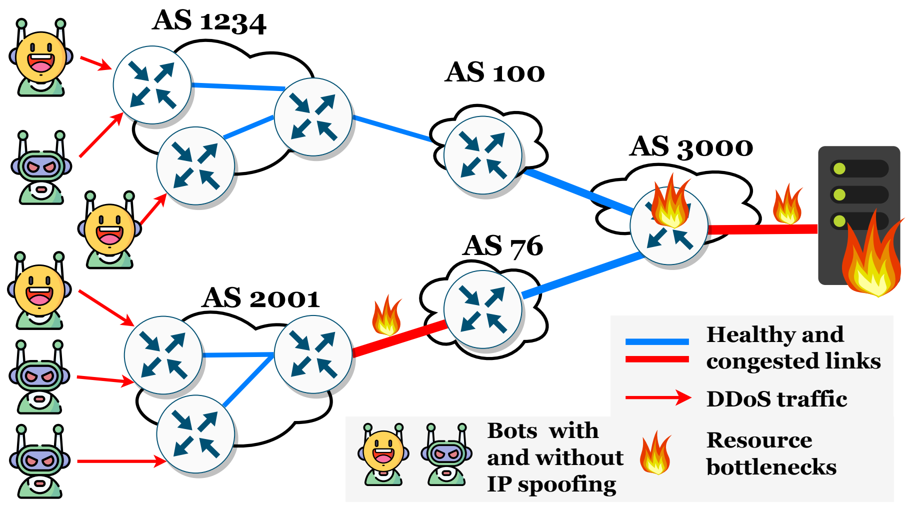


Figure 1. The resources an attacker can target on the Internet.

attacker disrupts any critical path of the service delivery process, the service is then crippled. DDoS is only not about overrunning the servers that host services. Figure 1 illustrates what and where these resources are on the Internet at a high level. We broadly categorize the resources by their belonging networks (i.e., edge network resources and transit network resources) on the Internet.

2.1.1.1 Edge Network Resources. The Internet design determines that most services are hosted in edge networks. Such a network (e.g.enterprise and university networks) consists of many hosts that serve various services. In most cases, adversaries launch DDoS to occupy most if not all host resources of a service, which prevents the service from serving its clients. However, because an edge network's link bandwidth is commonly limited, an attacker may also overrun the bandwidth to prevent legitimate users from reaching all services hosted in the edge network. Hence, an attacker may overwhelm the following resources in an edge network:

- **Network resource**: An attack can congest the network link(s) of a service. It may be a link from a network provider to a campus network or within the campus network. For example, the link between AS 3000 to the host in Figure 1.

- **Host computational resource**: An attack can force a host to perform an overwhelming amount of tasks to prevent the host from providing its service to benign users. For example, the attack can send frequent and crafted HTTP requests to a web server to trigger the server to perform computationally heavy tasks.

- **Host memory and input/output (I/O) resource**: An attack can force a server to perform a vast amount of I/O operations to reduce the server performance. For example, the attack can frequently request unpopular data from a server to evacuate the popular data in the server's memory, rendering a high cache-miss rate.

2.1.1.2 Transit Network Resources. Transit networks have a higher network capacity than edge networks. However, their network capacity is a finite resource nonetheless. Today, a large botnet can easily overwhelm a transit network. Figure 1 illustrates the link congestion between autonomous system (AS) 2001 and AS 76 due to an overwhelming amount of DDoS traffic (in the rest of this work, we use *AS* and *network* interchangeably). Specifically, AS 2001's border router can no longer cope with the traffic towards AS 76. In other words, the router's packet buffer is overflowing and dropping packets.

However, network links are not the only attack target in transit networks; as more ASes adapting to the software-defined networking (SDN) design [66], attackers can exploit any vulnerabilities in the new design. For example, an attacker may launch a DoS against the controller-to-switch communication channel of an SDN network [216, 210] (discussed in Sec 2.2.2.2).

2.1.2 Attack Preparation. A botnet is essential to any DDoS attack. To build botnets, adversaries need to compromise as many vulnerable Internet-connected devices as possible. Unfortunately, the Internet is growing with more vulnerable devices every day, which means the adversaries are increasing the capacity of their botnets. Therefore, we continue to see large botnets over the years. Even if an adversary does not own any bots, he/she can easily rent bots from DoS-for-hire websites [28]. The botnet owners are monetizing their bots, which provides a strong incentive for them to compromise more devices. Finally, attackers may also rent low-cost virtual machines as their bots from hundreds of cloud providers [174] or exploit residential proxy providers [112] to hide real bot IP addresses behind residential proxy IP addresses.

Once an attacker has a botnet, he/she can command the botnet to generate attack traffic. A smart attacker often tailors the attack traffic to the attack target (i.e., conducting *reconnaissance* of the attack target) to maximize/minimize the attack damage/cost, e.g., to exhaust a service's host resources or the network resource. The smart attacker also evaluates whether he/she is capable of disabling the attack target. For example, the attacker may not attack a website that is protected by a content delivery network (CDN) unless the attacker has the confidence to bring down the entire CDN, which is often distributed geographically. Instead, the attacker will find out the original server that announces the website updates to the CDN and then attacks the original server.

Worse, an attack is more damaging when coupled with IP spoofing (as shown in Figure. 1), an infamous problem the network community has not to fully addressed. When a bot can spoof its (source) IP address with any IP address, it can send spoofed packets *directly* or *indirectly* to bring down its target. In the first case, the bot sends spoofed packets with any source IP addresses directly to its target. In the latter case, the bot sends a public service (e.g.DNS) with the spoofed IP address of its target's IP address, the public service will then reply packets (e.g.DNS response), often larger than the spoofed packets, to the target. The target in both cases does not learn the bot's true IP address; the difference is in the number of source IP addresses the target will see. In the first case, the target may see traffic from *all* possible IP addresses, while in the second case, the target sees IP addresses of a set of public services. Bots with IP spoofing capability allow them to spoof every bit of a packet, making the defense against their traffic challenging. Although anti-spoofing solutions exist (e.g.ingress filtering [52]), they require a high deployment rate to be effective. According to CAIDA's Spoofer project [22], more than 23% of ASes in the study allow IP spoofing. Note that the study includes $\sim 8,000$ participating ASes, which only account for 10% of all ASes on the Internet.

2.1.3 DDoS Attack Taxonomy. We classify DDoS attacks based on their **intended** resource targets, as shown in Figure 2. Specifically, an attack may disable a service by overrunning its 1) network resource or 2) host resource. The former prevents legitimate traffic from reaching the service, and the latter leaves little to no service capacity to its legitimate users. Note that we do not consider the absolute attack volume in our classification; an adversary can always use an attack designed to overwhelm host resources to overwhelm the service's network resource.

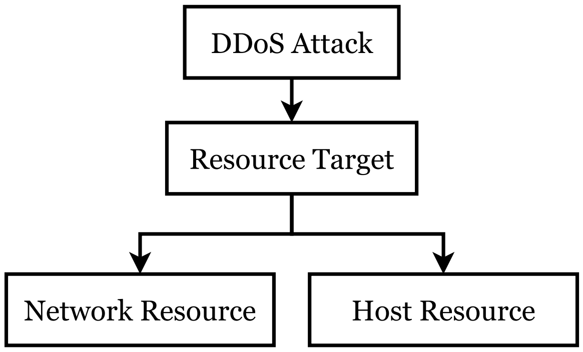


Figure 2. Intention-based DDoS attack taxonomy

2.1.3.1 DDoS on network resource. To prevent legitimate users from reaching a service, the attacker needs to send enormous traffic towards the networks that host the service. The attack traffic bandwidth needs to match, if not more than, the inbound network bandwidth of the service (e.g.the network link between AS 3000 and the server in Figure 1). When the attack kicks in, the benign clients that implement any congestion control algorithms will limit their traffic sending rate, hoping to mitigate link congestion. Moreover, suppose a botnet can generate traffic at a fixed bandwidth, the botnet may generate traffic at (1) a low packets-per-second rate with large packet sizes or (2) a high packets-per-second rate with smaller packet sizes. The former is prone to fill up a router's buffer space while the latter attempts to overrun the router's packet switching speed.

To the best of our knowledge, real-world attacks in this category are mainly different amplification attacks, also known as reflection attacks. As the name suggests, an amplification attack (e.g.DNS-based, memcached-based, CharGen-based) allows a botnet to amplify its total attack bandwidth. For example, in DNS amplification attacks [197], bots send DNS resolvers requests using spoofed source IP addresses. The spoofed addresses belong to the attack target (i.e., some service host or networks that carry traffic of the host). The resolvers then send responses, which often contain more data than the requests, to the spoofed IP addresses. The bots amplified their attack bandwidth in order to congest the network resources. Thus, while the attack consumes the DNS resolvers' host resources, the attack's intended resource target is the network resource of a service. Not all amplification attacks require IP spoofing. However, without IP spoofing, the requirement to launch such an attack is often challenging. For example, in 2015, an adversary injected a malicious JavaScript script into a top website, *Baidu.com* [17]. When users visited the website, their web browsers executed the script, sent attack traffic to the website *GitHub.com*. Due to the sheer number of users who visited *Baidu.com*, the script essentially turns the users into a huge botnet, allowing the adversary to amplify the attack. As a result, *GitHub.com*'s network resource was fully consumed by the attack.

The network community is familiar with the real-world attacks in this category. Specifically, bots do not follow the traffic pattern of benign clients and their packets contain obvious attack signatures. Therefore, we can easily detect and mitigate such attacks when given enough time to react. However, researchers discovered and proposed advanced attacks on network resources, such as those described in [195, 86] (reviewed in Sec. 2.2). Worse, the Internet started to see glimpses of these advanced attacks [30] in action. Thus, the efficacy of existing best practices is questionable against the advanced attacks.

2.1.3.2 DDoS on host resource. To occupy a service's host resource, the attacker needs to send an overwhelming amount of malicious requests to the service so that the service only processes the malicious requests. Because a service is built on top of multiple software abstraction layers (i.e., software stack), the attacker may target any layer to disable the service. Due to the variety of software stacks, attacks in this category are often more specific than attacks on network resources. In other words, the attacks in this category are not generally applicable to all hosts. For example, the attacker cannot effectively launch an HTTP-based attack (e.g.low-rate HTTP attack [212]) against servers that do not run HTTP services. However, because these are specific attacks against software, such attacks often require a smaller botnet than what is required for an attack on network resources. Some may not even need a botnet to disable a service. For example, a TCP SACK panic attack [166] only requires a few crafted packets to bring a host down.

In this category, there is one attack that continues to work effectively over the years: SYN flood attack [47]. In an SYN flood attack, bots send a targeted host a huge amount of TCP SYN packets. When the host receives such a packet, the host creates a TCP connection and waits for the packet sender to finish the TCP handshake process — the bots will never finish the process. Therefore, the bots force the host to maintain an increasing amount of half-open TCP connections. Because the host has a fixed memory space for storing TCP connections, the host will run out space to create new TCP connections, which prevents benign users from accessing the host's service(s). The SYN flood attack is interesting as it exploits the foundation of the TCP/IP network. Namely, a TCP server must accept SYN packets to establish the communication channels with its clients. However, the server cannot differentiate if an SYN packet is benign or malicious unless the SYN packet sender, which is identified by its source IP address, repeatedly offends the server. Unfortunately, source IP addresses are not reliable information to identify the senders, as we covered in Sec. 2.1.2. While many adversaries today use the SYN flood attack to overrun the network resource, the intended resource target of this

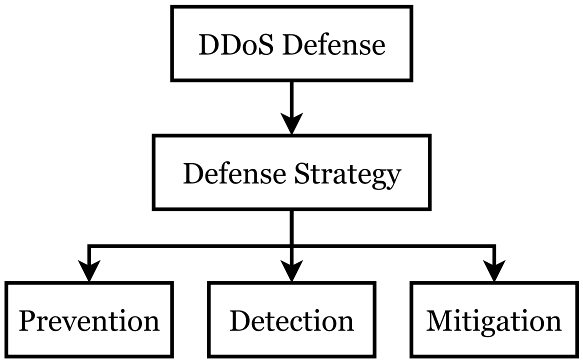


Figure 3. DDoS defense taxonomy based on strategy

attack by design is the TCP connection capacity of targeted hosts (i.e., the memory and I/O resource of a host).

2.1.4 DDoS Defense Taxonomy. This section studies DDoS defense solutions by their intended defense strategies (as shown in Figure 3): **prevention**, **detection**, and **mitigation**. We acknowledge that there are other defense strategies (e.g.attack attribution). However, we do not include them as they are challenging subjects that warrant separate studies. We describe each strategy's advantages and disadvantages (if any) when deployed at different network locations on the Internet (as illustrated in Figure 4). Specifically, we analyze solutions deployed at: 1) networks that originate attack traffic (denoted as **source-end**), 2) networks that receive attack traffic (denoted as **service-end**), and 3) networks that route attack traffic (denoted as **in-network**). Each network location works with different input data and has different defense capabilities.

2.1.4.1 Prevention. Solutions in this category prevent adversaries from launching **successful** DDoS attacks against services. Note that we cannot prevent a bot from sending packets towards a service; the Internet design allows any node to send arbitrary traffic to any other node. However, we can enhance the Internet with preventative solutions to reduce successful DDoS attacks. We summarize prevention solutions by their deployment locations on the Internet as follows.

Source-end prevention A source-end prevention solution is to 1) secure hosts to prevent adversaries from building botnets or 2) filter *known* malicious traffic proactively at source-end networks (e.g.networks can implement ingress filtering [52] to prevent spoofed packets from leaking into the Internet). These solutions require a high deployment rate to have a real-world impact.

Securing hosts is a collaborative effort between software vendors and users. First, the vendors should monitor for the security flaws in their software and provide software patches. Then, the users need to install the patches to secure their hosts. Unfortunately, the collaboration between vendors and users is often lacking in the real world. Worse, some vendors may not provide any software patches and leave many hosts running software with known vulnerabilities [8]. For example, a vulnerable home router may never receive a patch from its vendor.

Filtering known malicious traffic at source-end networks is mainly a deployment challenge. While some networks may not be aware of the prevention solutions, others *choose not to deploy* such a solution for monetary incentives [100]. For example, despite years of efforts, the Spoofer project [22] shows that roughly 25% of the ASes in its study allow IP spoofing. (Note that the study only covers a subset of the ASes on the Internet.)

Service-end prevention A service-end prevention solution is to serve the authorized users; it cannot prevent DDoS attacks that target a service's network resources. For example, a service may employ CAPTCHA [206] to differentiate humans from bots, and only allocate the service resources to humans than bots.

However, the allocation process itself consumes resources and DDoS attacks with thousands of bots can still overrun the service. Therefore, service-end prevention is inherently disadvantageous against DDoS attacks as the service has a limited amount of resources. Worse, smart attackers can find means to authenticate its bots to launch DDoS attacks. Despite its limited defense capability, service-end prevention solutions are critical to ensure a service does not waste its host resource.

In-network prevention An in-network prevention solution prevents DDoS from succeeding at transit networks of the Internet. In-network prevention solutions require a smaller deployment rate than source-end prevention solutions to be effective. Because in-network prevention solutions require significantly fewer deployments than source-end prevention solutions (i.e., the number of ASes that need to deploy a solution). Each deployment has a positive, extensive impact on DDoS defense. There are many approaches to implement such a solution. For example, one may employ a CDN to absorb DDoS traffic at multiple data centers close to source-end networks; if the DDoS traffic first traverses through the CDN, each CDN node can either return the requested information or reject the malformed traffic (i.e., TCP SYN flood traffic) preemptively. A service may also change its service locations on the Internet to dodge DDoS; the service may assign benign clients to a set of hosts that are not overwhelmed and direct ill-behaved clients to hosts that are already overwhelmed. Finally, some proposes new designs for the Internet to become more resilient against DDoS attacks.

2.1.4.2 Detection. DDoS detection is to determine whether a DDoS attack is occurring, and if so, identify the attack traffic (e.g.classify attack traffic by n-tuple packet header fields). The detection results are then used to mitigate attack traffic. The deployment location of a detection solution directly affects its

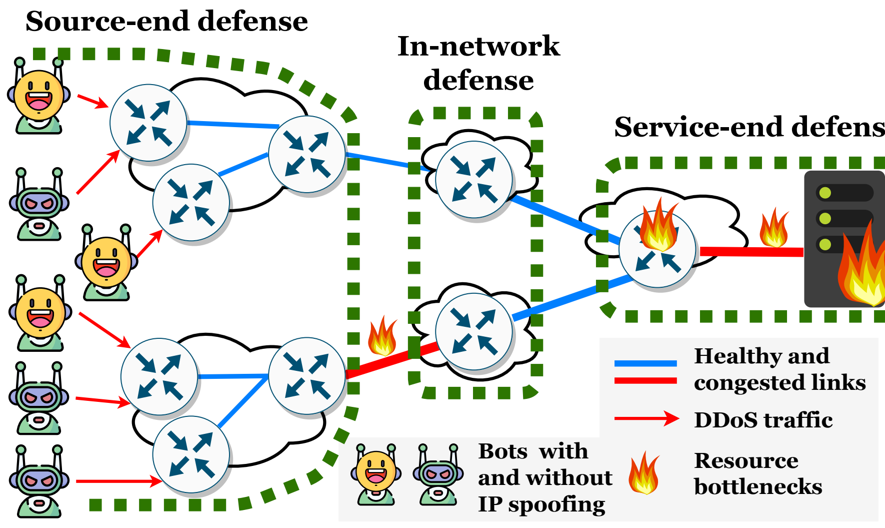


Figure 4. DDoS defense at different locations.

input data, thereby limiting the solution's capability. For example, source-end and service-end networks are both edge networks; the former detects DDoS with the *outbound* traffic towards other networks, and the latter detects DDoS with inbound traffic from other networks. Deployment locations also determine the amount of traffic a detection solution needs to compute, which affects the detection performance. For example, if a detection solution is not scalable for high-volume traffic, it may generate outdated detection results, and lead to ineffective defense.

Source-end detection Every source-end network contributes to a small percentage of a DDoS attack. Because the detection at such a location is exposed to partial attack traffic, it can only produce local-optimal results. Fortunately, for most real-world DDoS attacks, the traffic at each source-end network is sufficient for detecting significant outliers that behave abnormally. For example, such solutions are suited for detecting SYN flood attacks, amplification-based attacks.

The cost to run a source-end detection solution is low per network; the total detection load is distributed across all source-end networks. Source-end networks may also exchange their detection information to improve its detection results.

Service-end detection The service-end detection solutions have access to most of the DDoS traffic from *common* real-world DDoS attacks. Therefore, such a solution can leverage the attack traffic to produce more accurate detection results. Furthermore, the solution may even leverage host-level information of different services to improve its detection results. Unfortunately, as adversaries discover new attacks, service-end networks may not always see all the attack traffic. In other words, service-end and source-end networks can only produce local-optimal detection results in such attack.

In-network detection In-network detection solutions share to a lesser extent the advantages and disadvantages of source-end and service-end solutions. For example, each transit network helps to forward traffic from many edge networks. Therefore, the transit networks have a better chance to capture DDoS traffic but a less complete picture than a service-end solution in common DDoS attacks. Meanwhile, because transit networks carry more traffic than edge networks, it is more costly to run DDoS detection than edge networks.

Note that if a transit network is the only upstream network of a service-end network, the transit network has a better view of the attack traffic than the service-end network. It is because the transit network can capture the dropped packets on the link to the service-end network. However, suppose the transit network is not the only upstream network of the service-end network. Then, the detection solution needs to overcome the challenge of asymmetric Internet routing. Specifically, the detection solution can only use one direction of a communication to detect DDoS traffic, which can severely affect the detection accuracy.

2.1.4.3 Mitigation. A DDoS mitigation solution is to lessen DDoS traffic towards services. While both prevention and mitigation solutions aim for removing attack traffic, they differ in their operational models. A prevention solution uses pre-determined policies to filter traffic that is (obviously) malicious *proactively*; it removes attack traffic with certainty. However, a mitigation solution relies on a detection solution's results to mitigate the attack *reactively*; it may remove benign traffic and may not remove some attack traffic since detection solutions produce false positives and negatives. Moreover, a mitigation solution's deployment location dictates its capability.

Source-end mitigation A source-end mitigation solution can filter DDoS traffic by attack signatures, impose bandwidth limits on abnormal traffic, or allocate clients fair shares of the network bandwidth. Such a solution incurs a low traffic computational overhead but requires a high deployment rate. Note that a source-end mitigation solution receives mitigation requests from other networks (i.e., service-end networks), such a protocol is subject to exploitation by adversaries (a high risk feature for edge networks to implement and maintain). Therefore, the solution needs a secure protocol to authenticate and verify each request.

Service-end mitigation Similar to service-end prevention solutions, it is often too late to mitigate DDoS attacks at the service end. For example, an attack may congest the network resource of the service, or the service runs out of its host resource to function. In the former case, the service cannot mitigate the attack locally. In the latter case, with proper local traffic mitigation, the service can continue to provide its benign users a limited but functioning experience.

In-network mitigation Transit networks are the most common DDoS mitigation location to defend against large-scale attacks. A transit network (e.g., a service's upstream ASes) can mitigate attacks as long as (1) its ingress links are not congested and (2) it has a sufficient amount of mitigation capacity. If the transit network is overwhelmed, it needs to rely on its upstream ASes to mitigate the attack. Finally, similar to source-end mitigation solutions, in-network mitigation solutions also receive mitigation requests from other networks. Therefore, such solutions should also ensure that the mitigation requests are properly verified.

2.1.4.4 Summary of DDoS defense strategies. We covered the advantages and disadvantages of three DDoS defense strategies, i.e., prevention, detection, and mitigation. Each strategy may be deployed at different network locations (source-end, service-end, and in-network). Because each strategy and network location have unique advantage and disadvantages, there is no one-size-fits-all defense against all DDoS attacks. For example, an in-network mitigation solution cannot fight off all DDoS attacks efficiently without the accurate detection results from service-end networks.

2.2 Advanced DDoS Attacks

Real-world DDoS attacks often employ simple, practical techniques, and the defense methods against them are well known. We see that this attack trend continues in many incident reports from DPS providers. Lately, we see real-world attacks that employ advanced attack techniques described in academic research. These advanced attacks: (1) do not directly flood targeted services but the network resource of the services; exemplified by the Coremelt attack [195] and

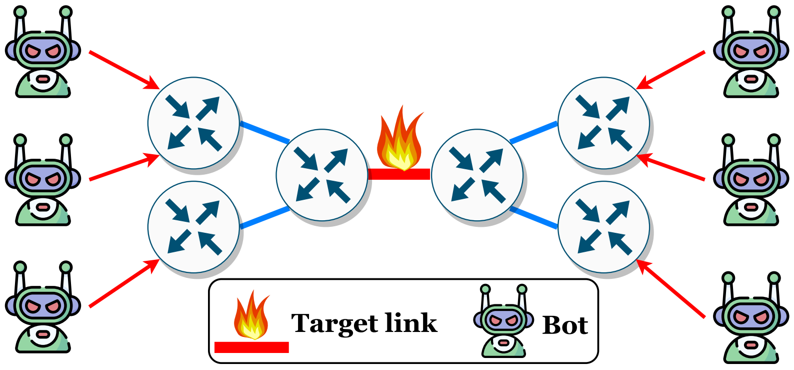


Figure 5. The Coremelt Attack [195]

the Crossfire attack [86], or (2) do not flood the targeted services constantly but deceive legitimate hosts to reduce their service request frequency; exemplified by the pulsing attacks [196, 88, 175]. Below, we first qualitatively analyze the advanced attacks by their (1) defense difficulty and (2) launch requirements. We then summarize both common and advanced attacks with the above qualities in Table 1.

2.2.1 Attacks on Network Resource.

2.2.1.1 The Coremelt Attack. Studer et al. [195] proposed the Coremelt attack in 2009, an attack that aims to overwhelm transit links on the Internet; it does not target any particular service but all services that rely on the overwhelmed transit links. In this attack, bots exchange 'legitimate' traffic among themselves, aiming to overwhelm the transit links that carry their traffic. The traffic is 'legitimate' since all traffic among the bots can follow application protocols. Furthermore, the bots could mimic the traffic patterns of benign users to mask the attack. Hence, the attack is both difficult to detect and mitigate. Figure 5 illustrates a high-level example of a Coremelt attack.

However, the requirement to launch this attack is questionable on today's Internet. The work uses inferred AS-level topologies [25] and a simplified link bandwidth model to evaluate the attack. It does not consider the rich, interconnected, router-level paths in transit networks nor the load balancing schemes that utilize these rich connections. The work also assumes that there exist botnets that can generate sufficient 'legitimate' traffic to overrun core transit links. While not impossible, it is still extremely challenging to realize since most DDoS attacks today produce less than 100 Gbps attack bandwidth. Finally, the attack does not target any specific service, which reduces its practical value. Nevertheless, the work influenced another attack (i.e., the Crossfire attack [86]) on network resources that is more practical to launch.

2.2.1.2 The Crossfire Attack. Kang et al. [86] proposed the Crossfire attack in 2013. The attack sends traffic towards hosts in a selected set of networks to overwhelm their shared network resources. Unlike common DDoS attacks that send all attack traffic to the targeted victim services, each bot in the attack distributes its traffic to attack multiple public-facing hosts who share the intended victim services' network resource. Therefore, the Crossfire attack has a more *focused target area* than the Coremelt attack [195].

The attack establishes low-volume, 'legitimate' connections with a set of public-facing servers of the selected ASes that share the same upstream network resource. Next, it collects traceroute results from each bot to the target area — a set of edge ASes that share the same upstream ASes. It then uses the results to identify a set of persistently appeared transit links to target. Finally, the attacker commands the bots to send 'legitimate' requests to the public-facing servers above. With 'legitimate' requests, the bots can overwhelm the link bandwidth of the targeted transit links. An example Crossfire attack is shown in Figure 6.

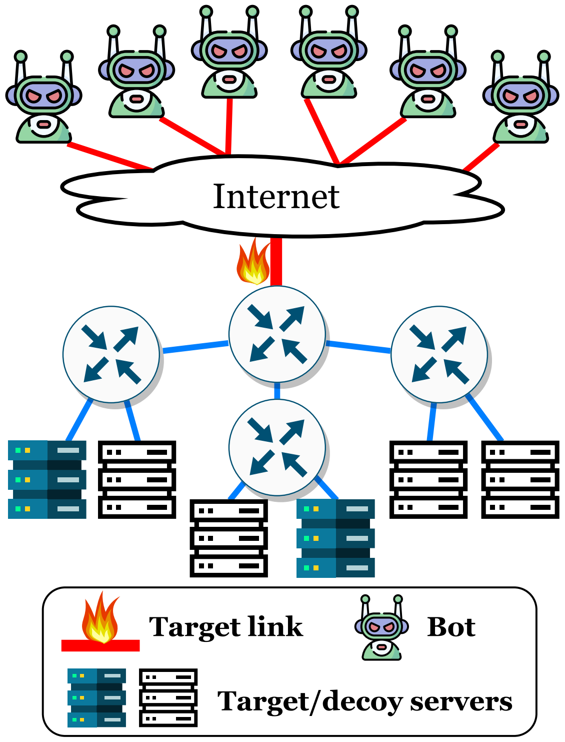


Figure 6. The Crossfire Attack [86]

Note that during a Crossfire attack, each attacked network only receives partial attack traffic and each attack flow is low volume. The authors claim that real-world intrusion detection systems (IDS) will fail to sound alarms. Therefore, the DDoS mitigation will not be initiated. While the Crossfire attack requires less attack resource than the Coremelt attack, the resource requirement is not trivial as it needs to overrun multiple links of transit networks that carry the targeted services' traffic. The authors also assume upstream networks do not run anomaly detection systems, which may detect the attack with the help of their customer (i.e., downstream networks).

2.2.2 Attacks on Hosts.

2.2.2.1 Pulsing Attacks. As shown in Fig. 7, a botnet sends frequent and bursty attack traffic, defined as attack pulses, to a targeted service in a pulsing

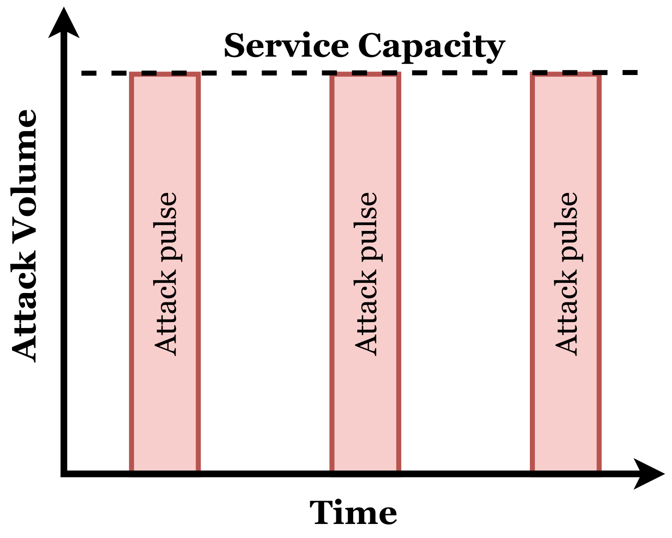


Figure 7. A Pulsing Attack Example

attack [196]. Since the length of each attack pulse is short, each pulse only disables the service temporarily; this attack pattern is different from traditional DDoS attacks where the attack traffic constantly occupies the targeted resources. A pulsing attack can trick benign hosts into believing in congestion with frequent attack pulses. As a result, the legitimate hosts reduce their service request rate hoping to alleviate the congestion. A pulsing attack may not trigger DDoS detection solutions to generate alerts; many detection solutions may use coarsely-grained network information (i.e., NetFlow and sFlow) that does not capture attack pulses.

While many pulsing attacks, such as those in [99, 113, 164], have been proposed, they are challenging to launch for two reasons: (1) bots are not synchronized to generate attack pulses, and (2) they require a centralized controller that coordinates the bots. Ke et al. [88] presented CICADAS to overcome the issues above. CICADAS implements (1) a decentralized attack system that uses link congestion as an indication to synchronize attack pulses and (2) a Kalman-filter-based algorithm for each bot to estimate when to send attack traffic. CICADA involves the following steps to launch a pulsing attack against hosts who produce congestion-aware flows:

> Bootstrap: the attacker first initializes each bot with a set of attack parameters, e.g., which link carries the traffic of targeted service, how bursty the pulse should be.
>
> Lurk: Each bot probes the target link to estimate the round-trip time when there is no attack.
>
> Attack: bots rely on the measured round-trip times above to infer link congestion and use link congestion as the signal for synchronizing their attack traffic, producing attack pulses.

As a result, the bots frequently congest the network link of the targeted service. The legitimate hosts reduce their traffic sending rate as they are tricked into believing network or service congestion. The authors evaluated CICADAS in an emulated environment. They show when the duration of each attack pulse is set to 200 ms, CICADAS can effectively reduce the throughput of benign TCP flows to as low as 30%.

Shan et al. [175] proposed a low-rate pulsing attack against web applications. The attack aims to force the web applications to incur long-tail latency — a small percentage of users experience a service delay for longer than the average service delay. Unlike common low-rate attacks that create *long-lasting* connections with victim servers and waste their memory or I/O resource, the attack [175] creates *short-lasting* resource contentions on web applications. The attack exploits the design of many web applications. A web application comprises a web server, a backend program that handles the business logic, and a database that saves state information. Each component above can be hosted on a different host. The attack sends a combination of bursty HTTP(s) requests to a targeted web application and employs Kalman filters to find the requests that generate predictable short resource contention periods on the web application. As a result, the application has to drop or queue requests from legitimate users, introducing service latency. In their experiment, the authors show that the attack can cause more than a second of service delay at the 95th percentile. Because each resource contention period is short, the authors claim that existing, coarse-grained anomaly detection systems cannot find the attack since the attack does not entirely overwhelm the service's host resource. Therefore, the attack can impose long-term damages to the service quality of targeted web applications.

2.2.2.2 DoS in software-defined networking (SDN). Today, network operators may program their networks via network controllers (e.g., Ryu [142], ONOS [198]); such networks are referred to as software-defined networks. These network controllers play a critical role in deciding how networks process their packets. However, these software-defined networks are susceptible to DDoS attacks if poorly designed [216].

Wang et al. [210] proposed an attack to exhaust the I/O resource of network controllers. This attack is viable when the victim SDN network employs a reactive-forwarding scheme. In a reactive-forwarding scheme, a packet switch may not know how to process a received packet and the switch must query their network controllers to learn how to process the packet. Thus, the attack is to send an overwhelming amount of packets that the switch does not know how to handle. The switch then sends a query for each received packet to its network controller.

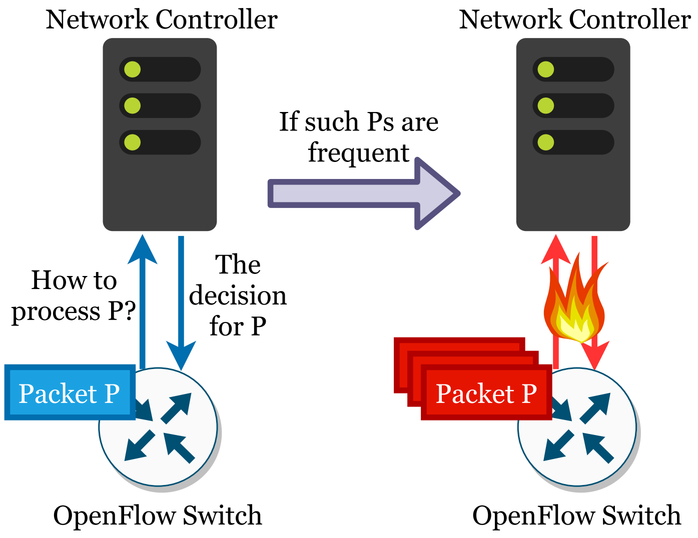


Figure 8. DoS in SDN

Table 1. Selected DDoS attacks

	Attack Target	Example	Launch Difficulty	Defense Difficulty	
				Detection	**Mitigation**
Common Attacks	**Network resource**	Amp. attacks [45, 197, 143, 124]	Easy	Easy	Easy
	Host resource	Low-rate attacks [74, 167]	Easy	Medium	Medium
		Layer 4/5 flooding [38, 39, 47]	Easy	Medium	Medium
Advanced Attacks	**Network resource**	Coremelt [195]	Hard	Hard	Medium
		CrossFire [86]	Medium	Hard	Medium
	Host resource	CICADAS [88]	Medium	Hard	Medium
		Tail Attack [175]	Medium	Hard	Easy
		DoS on SDN [210]	Hard	Medium	Easy

As a result, the switch is forced to inundate the I/O channel between itself and the controller. Figure 8 illustrates such an attack. Once the channel is overwhelmed, the switch will quickly fill up its packet buffer as it waits for the decisions for the queries, and eventually, the switch has to start dropping packets.

2.2.3 DDoS attack summary. We summarize common and advanced DDoS attacks in Table 1. While most common attacks are easy to launch, they are also easy to detect and mitigate as they consist of discernible attack signatures. Meanwhile, network security researchers continue to discover advanced DDoS attacks that are more stealthy than the common ones. DDoS victims can mitigate some advanced attacks locally once detected (such as described in [175, 210]), the victims cannot mitigate other attacks [195, 86, 88] as their upstream network resource is overwhelmed. Worse, attack traffic may not contain signatures that mitigation solutions can leverage, increasing the defense difficulty. The advanced attacks are, however, difficult to launch as they (1) require a high attack budget [195, 86], (2) rely on accurate network measurement results during the attack [88, 175], or (3) have access to specific network environments [210].

In the real world, we see DDoS attacks such as carpet bombing [30] that leverage the practical techniques from the advanced DDoS attacks (e.g.Crossfire [86]). While such attacks remain easy to detect, they challenge the capacity and capability of existing DDoS defense solutions on the Internet. We expect to see more DDoS attacks leverage multiple attack techniques from attacks such as Crossfire [86] and CICADAS [88]. More importantly, we need to understand the damage such an attack can impose on the Internet. For example, if an adversary composes a new DDoS attack with pulsing and the Crossfire-like attack technique, how many services or edge networks can the adversary pin down simultaneously? Are state-of-the-art defense solutions ready for such an attack?

CHAPTER III

RELATED WORK

In this chapter, we study the DDoS defense research from the past two decades and summarize the challenges for each strategy. Specifically, Sec. 3.1 reviews a wide range of prevention solutions that are designed to enhance the Internet architecture to become resilient against DDoS attacks. In Sec. 3.2, we cover different detection solutions and checks if their designs and the corresponding experiments are sound to detect the advanced attacks. Finally, we examine mitigation solutions in Sec. 3.3. The mitigation solutions leverage DDoS detection results to filter the attack traffic. The section outlines the main mitigation models and lists the issues behind each model.

The content in this chapter appeared in the following unpublished work: Shi, L.; Li, J., Two Decades of DDoS Attacks and Defenses. Lumin Shi is the leading author of this work and responsible for leading all the presented analyses.

3.1 DDoS Prevention

In DDoS defense, prevention solutions are to either prevent attacks from happening or reduce the number of successful attacks. Given the variety of DDoS attacks, there is no silver bullet in DDoS prevention. We should treat each DDoS prevention solution as a building block to construct a more DDoS-resilient Internet. In this section, we classify and review well-received prevention solutions based on their deployment locations, as we discussed in Sec. 2.1.4.

The main ideas of the reviewed prevention solutions are as follows. The **source-end prevention solutions** are to:

- Secure hosts to prevent adversaries from building large botnets or

- filter malicious traffic proactively at edge networks, such as spoofed traffic.

Next, the **service-end prevention solutions** are to:

- Authenticate clients to prevent attackers from abusing clients' resources or
- rate limit traffic at the network or application layer based on the normal traffic profile.

Finally, the **in-network prevention solutions** are to:

- Absorb DDoS traffic with dedicated network infrastructure,
- dodge attacks by shifting the location of a service, or
- redesign the Internet architecture to become resilient against DDoS attacks.

3.1.1 Source-End Prevention.

3.1.1.1 Host Security. As the first line of defense, host security affects the scale of DDoS attacks. Standard practices, such as enabling host firewall and keeping the software up to date, are critical to host security. However, with zero-day exploits, good host security practices alone cannot prevent adversaries from compromising hosts. Worse, the standard practices are challenging to implement in the real world; it is a collaborative effort among software vendors and their users. For example, in 2016, a Mirai botnet took down Dyn's DNS service, rendering many major services inaccessible to the public. The botnet mainly consisted of Internet-connected devices (e.g.home routers, security cameras) accessible via dictionary access credentials [8]. Some Mirai-infected devices come with no software support. While users can patch some of their devices, not all users are aware of the situation. Therefore, many Mirai-infected devices continue to impose threats to others even in 2021 [218].

3.1.1.2 IP-Spoofing Prevention. Today, DDoS amplification attacks continue to impose damage to many networks. Compromised hosts with IP spoofing capability are the primary enablers of such attacks. Therefore, IP-spoofing prevention is critical to address the attacks. Ingress/egress filtering [52, 91] remains the most effective and easy-to-deploy solution to prevent hosts from spoofing IP addresses. The solution requires edge networks to install filters that drop outbound traffic with source IP addresses that do not belong to them. It requires a high deployment rate at edge networks to prevent large-scale IP spoofing. Network routers from major vendors all have the solution built-in, and it only takes several minutes to enable ingress/egress filtering. However, the deployment of such a solution remains low on the Internet [22]. While IP-spoofing prevention solutions for transit networks exist, such as those defined in [153, 130], they require inter-AS collaborations that challenge their deployment difficulty.

3.1.2 Service-End Prevention.

3.1.2.1 Authentication. Services should only process requests made by human users than bots to prevent the bots from wasting the service's resources. Solutions such as CAPTCHA [206] require human users to complete a set of puzzles to help a service to differentiate human beings from bots. The puzzles are computationally expensive for bots to solve.

To provide a better user experience than CAPTCHA, Gummadi et al. [65] presented a human activity attester to authenticate legitimate users. The attester models a human's keyboard and mouse activity and only accept the requests that fit the model. In addition, the work relies on a trusted platform module (TPM) [201] to ensure that a malicious program cannot fabricate the activity data. While authentication solutions increase the attack difficulty, resourceful attackers can authenticate their bots nonetheless. Moreover, such a solution does not prevent adversaries from attacking the service's network resources.

3.1.2.2 Rate Limiting. A service may also limit each user's service request rate to prevent the scenario where a few users occupy all service resources (e.g., the service may set a maximum resource threshold each user can utilize). In 1999, Spatscheck et al. [192] proposed Escout, an operating-system-level solution that measures the amount of resource spent on each user request at different system layers. For example, if a web server needs to process an HTTP request, Escout tracks the computation, memory, and I/O utilization to process the request at different system layers. Escout allows the system owner to apply rate-limiting policies to assign fair shares of resources to each request. Note that service-end rate limiting solutions are inherently disadvantageous as they are subject to state exhaustion. (One may deploy rate limiting solutions at source-end networks. However, the challenge is to find a suitable service-specific traffic rate for each host at source-end networks. Moreover, given the number of services on the Internet, such a solution is difficult to scale.)

3.1.3 In-Network Prevention.

3.1.3.1 Absorbing DDoS Traffic. A DDoS attack is successful when it occupies a significant amount of resources from the targeted service. Thus, it is an arms race between attackers and defenders, and whoever possesses more resources wins. In other words, a DDoS attack is defeated if the attacked service can simply absorb the attack traffic without affecting its normal service level. For the service to absorb attack traffic, it needs to increase its resource capacity (i.e., network and host resources). The most accessible approach is to rent resources from cloud providers. For example, one can deploy a web service at multiple cloud providers which increases its total resource capacity. However, there is a challenge to address: how do we ensure that all service replicas have the latest information (i.e., data consistency) during a DDoS attack? Indeed, if the attack overwhelms the original host that sources information, the replicas can not fetch the information from the host. Cloud providers utilize both DNS and BGP to disperse the traffic towards the host among its replicas to overcome the challenge. Ideally, only the replicas can communicate with the original host that sources information. Once the traffic is correctly dispersed, each service replica only processes a subset of the service's traffic. Better, each replica can deny malformed traffic, often close to the traffic source network(s).

When using DNS to disperse traffic, such as the solution by Freedman et al. [56], each DNS server provides users (including bots) a list of the service replicas' IP addresses for them to fetch information from. The IP address of the original service host must be kept as a secret. Otherwise, the attacker can directly send attack traffic to that IP address. However, researchers in [80] show that the adversaries can often find the secret IP address to bypass replicas and attack the service host that sources information directly.

To address the limitation in the DNS approach, cloud providers today often use BGP to disperse the traffic among the service replicas; the technique is often referred to as BGP anycast by the network community. With the anycast technique, we no longer hide the IP address of the original service host. Instead, multiple service replicas can use the original service host's IP address. Specifically, a cloud provider sends out BGP announcements from multiple data centers on the Internet, and each announcement contains an attractive metric to reach the service host's network. As a result, the neighboring networks that receive the announcements are likely to choose the cloud provider to forward the traffic. *Note that not all neighboring networks will choose the announcement and change how they route traffic; some may have a fixed routing table.* Once the traffic is diverted, the cloud provider can route the traffic to its service replicas.

Figure 9 illustrates an example of how two data centers (i.e., X and Y) leverage BGP anycast to disperse traffic towards the original service host. Specifically, after BGP anycast is deployed, router C no longer carries the traffic for the host. The reverse proxies within X and Y process all the traffic (service requests). If a proxy has cached the requested data, it can immediately return the cached data without involving the original service host. The proxy can also drop the service requests if they are malformed.

Gilad et al. [62] presented CDN-on-Demand, a system that dynamically adjusts the number of service replicas at cloud providers to absorb DDoS or flash crowd traffic. The authors show that it incurs only a few U.S. dollars per month, which is considerably cheaper to commercial solutions. However, the work does not include its operational and maintenance costs that are significant to small companies. FastRoute [53] combines both the DNS approach and the BGP anycast approach to disperse networks traffic among service replicas. FastRoute first utilizes BGP anycast to attract network traffic close to its data centers. Then, within each data center, FastRoute uses DNS to further balance the network traffic. Finally, a FastRoute data center withdraws the BGP messages when it is overloaded.

3.1.3.2 Dodge Attacks by Shifting Service Locations. While services can rent more resources to absorb attack traffic (Sec. 3.1.3.1), not all services can or are willing to pay for more resources. For such a reason, researchers proposed various moving target defense (MTD) solutions [77, 78, 204, 42, 176] so

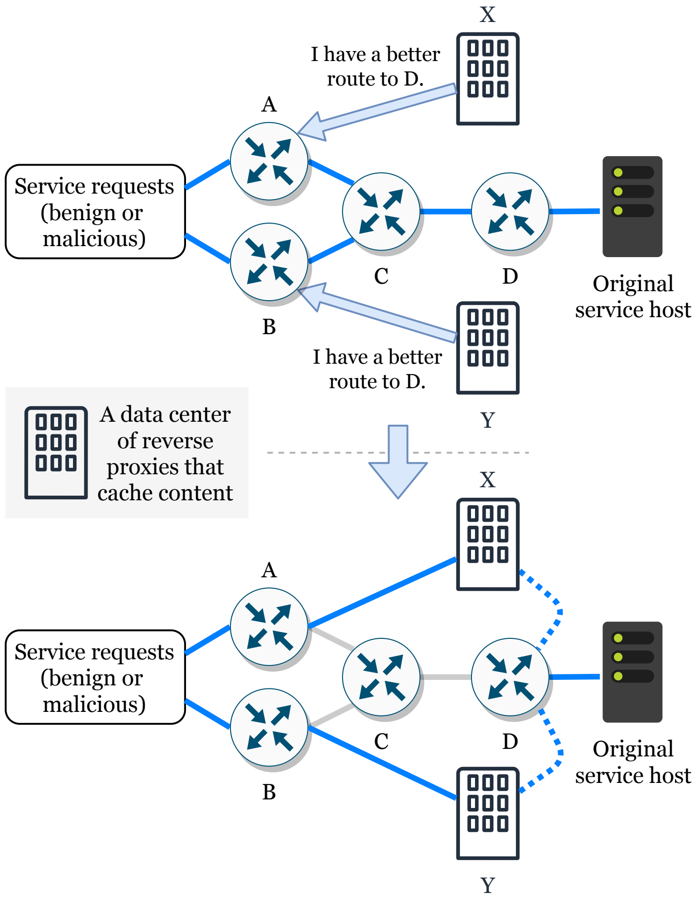

X
I have a better route to D.
A
Service requests (benign or malicious)
C
D
Original service host
B
I have a better route to D.
Y
A data center of reverse proxies that cache content
X
A
Service requests (benign or malicious)
C
D
Original service host
B
Y

Figure 9. Absorb attack traffic with the help of BGP anycast.

that the services can dodge attacks without needing to match the attack resources. The high-level goal of MTD solutions is as follows: Given a set of service replicas, an MTD solution isolates malicious clients onto a small set of replicas while serving the legitimate clients with the remaining/majority of the replicas. The key idea of MTD is to leverage group testing theory to isolate malicious clients (i.e., clients who use an abnormal amount of service resources) from legitimate clients [90].

MOTAG [77] is a system that defends against attacks on network resources. In MOTAG, each client first authenticates itself to obtain a proxy IP address (only known by the authenticated users). The clients then communicate with a protected service through the proxies. Whenever MOTAG detects a proxy's network resource

Table 2. Reviewed DDoS Prevention Solutions

	Main Idea	Example	Performance	Scalability	Deployability
Source-end (Sec. 3.1.1)	Host Security	Out of scope	N/A	N/A	N/A
	IP-Spoofing Prevention	Ingress filtering [52]	Excellent	High	Medium
		Park et al. [153] Mirkovic et al. [130]	Fair	Medium	Low
Service-end (Sec. 3.1.2)	Authentication	CAPTCHA [206]	Good	High	High
	Rate limiting	Escout [192]	Good	High	Medium
In-network (Sec. 3.1.3)	Absorb Attack	OASIS [56] CDN-on-Demand [62]	Good	Medium	High
		FastRoute [53]	Excellent	Medium	High
	Dodge Attack	MOTAG [77] Khattab et al. [90]	Fair	Medium	Medium
		Jia et al. [78] Debroy et al. [42] Shan et al. [176]	Fair	Low	Medium
	Patch/redesign Internet	Phalanx [46] ScoreForCore [84]	Fair	Medium	Low
		WebSOS [132] Mayday[7] SOS[89]	Good	Medium	Low
		Stavrou et al. [193]	Good	Low	Med
		STRIDE [70] SIBRA [14]	Good	High	Low

is overloaded, the system directs its legitimate clients to proxies that are not overloaded and keeps malicious clients to proxies that are overloaded. MOTAG repeats the process above until all malicious clients are directed to a small set of proxies. As a result, abnormal clients will experience a low service throughput. Jia et al. [78] proposed a system to improve MOTAG to address attacks on host resources. The work introduces a load balancer that monitors the host resource load of each proxy, and increases/decreases the number of proxies based on the host resource load.

Unfortunately, both MTD solutions above and other MTD solutions[42, 176] have strong assumptions. For example, they assume the attackers can not flood the authentication servers, which prevents clients from learning proxy IP addresses in the first place. Adversaries can also employ some of their bots to harvest the proxy IP addresses that serve legitimate clients if the bots behave nicely to the proxies, as pointed out by Venkatesan et al. [204].

3.1.3.3 Redesign the Internet. Because the Internet was not designed with security in mind, it enables DDoS attacks by default. Therefore, researchers have proposed solutions such as those defined in [7, 214, 89, 46, 84] to address the security problems of the Internet. The main idea of such solutions is to build an overlay network on top of the existing Internet and allow only the authenticated traffic to reach their destination networks.

Yaar et al. [214] proposed SIFF in 2004, a system that divides network traffic into privileged and unprivileged classes; the privileged class has access to higher network bandwidth. In a SIFF network, a client first needs to authenticate itself with a service host. The SIFF network will only forward traffic to the service host after the client obtains an access token from the service host. The token also puts the client traffic into the privileged class.

In the same year, Keromytis et al. [89] proposed *SOS* to prevent DoS attacks proactively. SOS consists of servlets (i.e., network proxies) that protect services. The system only allows servlets with secret, approved IP addresses to access the services. For a client to communicate with the service in SOS, the client has to authenticate itself with one of the servlets first. The client can then forward its traffic to the server via a servlet. Because SOS servlets proactively filter unauthenticated traffic, a DDoS attack cannot directly overwhelm the network resource of services in the SOS network. Later, WebSOS [132] was proposed as a system that integrates CAPTCHA [206] and SOS [89] to protect web servers from attacks. However, SOS does not prevent an adversary from learning the servlet IP addresses. As a result, the adversary can spoof the servlet IP addresses and send the spoofed traffic to the protected server directly (i.e., an insider attack).

Andersen [7] proposed Mayday, a system that improves SOS by separating SOS's overlay routing and filtering. The author finds that SOS [89] is susceptible to the insider attack. Mayday supports a set of overlay routing schemes to prevent attackers from learning the *approved* IPs. Mayday uses either 1) random routing or 2) doubly-indirect routing, which adds another routing layer in the overlay network to mitigate insider attacks. However, these additional protection mechanisms add both computational and network overhead to the system. For delay-sensitive services (e.g.Voice over IP (VoIP)), Mayday may not be their best defense solution.

Stavrou et al. [193] proposed the sweeping attack, an attack against the overlay-based prevention solutions. The authors also proposed an approach to address the attack. In the sweeping attack, bots attack a small set of proxies in an overlay network at a time. Thus, the targeted proxies' network resource is overwhelmed. The attacker frequently changes the set of nodes to attack so that the latency-dependent services will be severely affected (e.g.audio calls get interrupted every few seconds). This attack exploits the fact that the overlay nodes are composed of commodity machines with limited bandwidth and computational power. To address the sweeping attack, the authors proposed a spread-spectrum technique that requires the clients in an overlay network to send their packets across multiple proxies. The proxies then forward the packets to the protected server (the technique shares a similar concept as Multipath TCP [54]). Unlike Multipath TCP, the technique sends *duplicated* packets across multiple overlay nodes to ensure a high packet delivery rate during a DDoS attack. The technique works when the attacker cannot overwhelm *all* the overlay nodes simultaneously. However, this technique can adversely affect the Internet when it is widely used.

Namely, the effective bandwidth of a network link shrinks as the packet duplication rate increases.

STRIDE [70] and Sibra [14] are two prevention solutions to defend against attacks on network resources. Both solutions rely on SCION [227] to perform bandwidth allocations between different networks. (SCION is a new Internet architecture designed to address security challenges such as IP spoofing and route hijacking on the Internet.) Specifically, in SCION, a transit network negotiates its link bandwidth with another transit network on demand. End hosts in SCION must first create a short-term communication channel before they can talk to each other. These communication channels' bandwidth is negotiated by the end hosts and enforced by their network providers. However, at the time of this writing, the deployment of SCION is at its initial stage. It is too early to tell whether the network community will adopt its design.

3.1.4 DDoS Prevention Summary. While each of the three network locations (i.e., source-end, service-end, and in-network) has many DDoS prevention solutions, they are not comprehensive or cost-effective to defend against more advanced attacks (e.g., it is costly to build a defense system that absorbs attack traffic). In Table 2, we qualitatively rank each reviewed prevention solution's performance, scalability, and deployability.

The solutions in Sec. 3.1.3.1 are to absorb/trap attack traffic in regional networks before the attack traffic accumulates at the service-end networks. They deliver good to excellent performance in DDoS prevention. Moreover, they do not require additional collaboration from other ASes to work. Therefore, their deployability is high. However, their scalability is questioned in protecting all services or networks on the Internet. For example, cloud providers may not be able to absorb traffic for 1,000s of edge networks simultaneously, and the use of careless BGP anycast could impose an increased network footprint on the Internet. Indeed, Morau et al. [135] presented a preliminary study of the collateral damages caused by BGP-anycast-based defense solutions.

MTD solutions (Sec. 3.1.3.2) allow services to combat DDoS attacks with limited resources and regionalize the attack traffic via group testing. However, all reviewed MTD solutions have strong assumptions that are difficult to implement in the real world. Specifically, they do not consider the cost of migrating application states from one host to another, and they assume attackers cannot inform bots of the servers that serve benign clients. For these reasons, we value their performance as fair. We rank their scalability as medium as they require each overlay node to maintain states for each client.

The solutions in Sec. 3.1.3.3 propose new designs to make the Internet more resilient against DDoS attacks. From the technical perspective, they achieve their design goals and are scalable solutions. However, such solutions face deployment challenges in the real world. For example, real-world networks may consider the technical soundness while also considering the cost to implement and maintain the solution. Furthermore, for such a solution to work, it needs a significant number of ASes to participate, which is not only a technical challenge but also an incentive problem for the network community to overcome.

While it is virtually impossible to prevent all DDoS attacks from succeeding, we need to deploy various DDoS prevention solutions to reduce the number of vectors that adversaries can exploit. It is a collaborative effort from multiple parties. Namely, the manufacturers need to produce devices with long-term software support in host security, users should be aware of the security

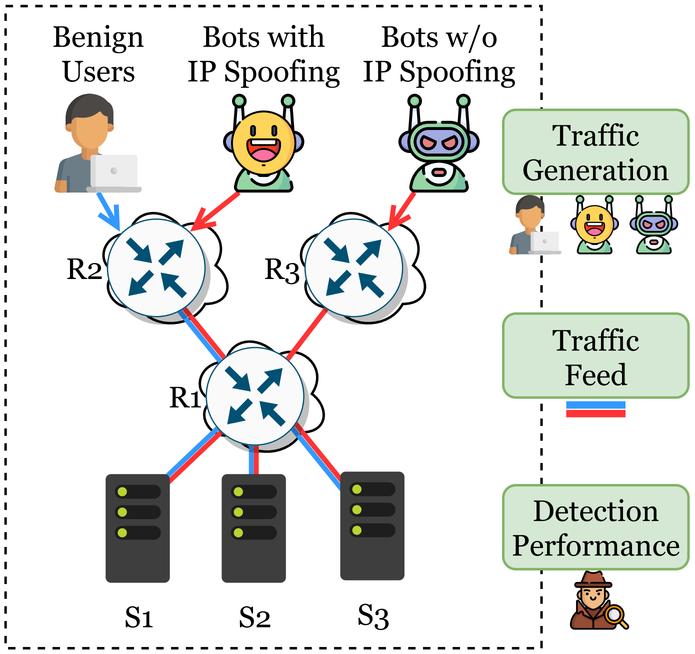


Figure 10. Evaluation factors of a DDoS detection system

vulnerabilities of their devices, and our network community needs to implement the best practices of DDoS prevention.

3.2 DDoS Detection

If a service cannot prevent a DDoS attack, the service needs to detect (and classify) the DDoS attacks before mitigating the attack. We discussed the trade-offs of deploying a detection system at different Internet locations in Sec. 2.1.4.2. Therefore, in this section, we review detection solutions by their deployment locations. We will find that most detection systems are designed for in-network or service-end deployment. Note that a DDoS defense solution may include a detection component and a mitigation component. In such a case, we strictly focus on the main idea of the detection component.

Figure 10 summarizes the important factors of a DDoS detection system. Specifically, we should realize that the system input (i.e., how traffic is generated) is critical to evaluating a DDoS detection solution. (e.g.does the traffic of benign users follow congestion control protocols?). Next, the traffic feed available to a detection system is not always ideal for detecting DDoS attacks. For example, the traffic feed at R3 is totally different from the traffic feed at R2. Worse, R3 may only see traffic towards R1 but not the other way around due to the asymmetric routing property of the Internet. Finally, the detection system must be efficient for its designed deployment location. In other words, is the system scalable to deploy in a high-throughput network? The detection speed is critical as DDoS attacks may change their behaviors over time; detection results are historical information and may not capture the attack traffic in real-time. Now, let us dive into the existing DDoS detection solutions with these factors in mind.

3.2.1 Source-End Detection. D-Ward [131] is a source-end DDoS defense system that detects and mitigates attacks. Every router in D-Ward tracks the statistics of each network flow (e.g., a TCP connection) and uses benign flow models to predict if a flow is benign or not. For example, if a TCP flow's inbound and outbound traffic rate lies outside the trained TCP traffic model, the TCP flow is then flagged as an attack flow. The obvious drawback of D-Ward is that it cannot detect attack flows that are within the benign flow models. In a large-scale DDoS attack, the attacker does not rely on a single bot but a botnet to carry out an attack. Therefore, the real-world effectiveness of D-Ward is questionable.

RAPID [125] is another source-end detection solution for IoT-based DDoS attacks. The work leverages machine learning algorithms such as multilayer perceptron (MLP) and long short term memory (LSTM) to detect malicious flows. RAPID aims to allow network operators to spend less time in manual parameter tuning, and claims that it is capable of porting a trained model to a new network environment. The work first evaluates network anomalies at a coarse-granular level, e.g., detect anomalies at /24 prefix level. If RAPID detects a certain prefix behaves abnormal, it zooms in on the prefix and evaluates network flows at the IP address level.

3.2.2 Service-End Detection. In many network anomaly detection solutions, such as those in [101, 191, 106, 181], they derive different static thresholds of a network using network information over a period of time (e.g.a threshold for the number of connections an IP address can have). These solutions raise alerts when the monitored network reaches one or multiple thresholds. The main concerns with these solutions are their accuracy and robustness. For example, if the threshold is set too low, they may produce an overwhelming amount of false positives, and they may not produce any alert if the threshold is set too high.

Hussain et al. [72, 73] and Jin et al. [79] proposed solutions that use the packet time-to-live (TTL) field to infer whether IP spoofing is involved in an attack. They assume that packets with spoofed source IP addresses will travel a different number of routers than those sent from the spoofed addresses. Therefore, spoofed packets and legitimate packets will have different TTL values. Unfortunately, such a scheme will not work when adversaries adjust the packet TTL values to mimic the TTL values of benign packets.

Gavrilis et al. [61] proposed a neural network DDoS detector that relies on three packet features to train the network: source port, TCP sequence number, and TCP SYN flag. Because it was an early work in 2005, the authors evaluate the detector against common flooding attacks (i.e., TCP SYN, UDP, and ICMP flood attacks). To understand what packet features in a network are more valuable for DDoS detection, Osanaiye et al. [150] proposed an ensemble-based feature selection method to choose the best feature set of a network for attack detection. The authors feed network traces to multiple filters (e.g., *information gain*, *Chi-square*), and evaluate the result from each filter to find the best packet feature set of the network. Unfortunately, the work does not provide details on the attacks it can detect.

Many entropy-based approaches [146, 217, 212, 15] use packet features, such as IP addresses, ports, and flow volume to tell attack traffic apart from legitimate traffic. Nychis et al. [146] presented a measurement study of entropy-based anomaly detection solutions, and the authors use information such as 1) IP addresses, 2) ports, 3) flow size distribution, and 4) the number of unique destinations that a host communicates. The authors used both synthesized network anomalies and traffic records from an educational network in their study. The study found there is a strong correlation between the number of connections per host and DDoS attacks. Xie et al. [217] proposed an entropy-based DDoS detection system to detect malicious HTTP requests. The system calculates the popularity of each accessed document and models temporal document access patterns during benign flash crowd events. It uses a custom hidden Markov model to compute entropy values based on the incoming HTTP requests. The authors evaluated the system by injecting DDoS attacks into a set of flash crowd event traces and validated whether the trained system can later detect those injected attack traces. The work is applicable to websites that host static web content, and it does not detect non-HTTP-based attacks. [15] is another entropy-based detection system, and the authors evaluated their solution against three small real-world datasets. They show the solution can differentiate the flash crowd events from the DDoS attacks. However, these small datasets are not representative of all networks, and they are composed of the most common DDoS attacks we see today.

Sun et al. [196] proposed a system to detect pulsing attacks on network resources. The system requires all upstream routers of a service to perform detection on their egress ports toward the service. The system frequently samples the network traffic (e.g., 100 times per second) and collects the time-series information of the network's link utilization to detect pulsing attacks. The system scans for the bandwidth utilization windows that are bursty and checks for whether there is a pattern of the bursty windows. If positive, the system identifies a pulsing attack. The cost of the detection is not trivial due to the high-frequency traffic sampling.

DDoS Shield [162, 163] contains a detection solution that detects attacks on host resources. The authors use HTTP servers as the evaluation test case. DDoS Shield tracks individual HTTP client's behavior metrics such as the request inter-arrival time and server utilization. DDoS Shield then calculates a suspicion value of each HTTP client based on the behavior metrics and triggers the mitigation process when necessary. Similarly, Feng et al. [51] proposed a reinforcement learning system to detect attacks on host resources. The work introduces a multi-objective reward function to guide the reinforcement learning agent to learn the suitable mitigation actions. The work shows a high detection efficacy in detecting malicious application requests in a controlled evaluation environment.

Afek et al. [2] proposed a method to extract traffic signatures from a DDoS attack. The work is to find a set of strings that only exist in network traffic when it is under DDoS attacks. However, the real-world efficacy of this work is challenged since the attack traffic can be either encrypted or randomized, which means there is no common signature of the attack.

3.2.3 In-Network Detection.

LADS [172] is an in-network defense system that detects attacks on network resources. The system relies on traffic information that is readily available in most networks: SNMP and NetFlow. LADS consists of a lightweight detection mode and a heavyweight detection mode. The lightweight detection mode monitors a network router's traffic volume, CPU utilization, and packet drops. It triggers the heavyweight detection mode when it finds anomalies. In the heavyweight mode, the system categorizes traffic into TCP SYN flows, ICMP flows, ICMP flows, etc. The system then checks the flow statistics against a set of *predefined thresholds* and alerts the network.

Zhang et al. [224] proposed a system to classify attack flows in pulsing attacks. The authors assume that the legitimate flows in a pulsing attack tend to reduce their traffic sending rate during the traffic congestion periods. The system acquires sampled packets to estimate the bandwidth utilization of a network's traffic flows based on the assumption. The system marks the flows that repeatedly appear in multiple congestion periods that do not reduce their traffic sending rates as attack flows.

Francois et al. [55] proposed Firecol, a defense system with an overlay network that forms protection rings around the subscribers (i.e., services). Each ring consists of nodes that can talk to each other, and the nodes exchange traffic information to detect DDoS attacks. Firecol employs a score-based attack traffic detection method. The system keeps track of 1) the volume of each network flow (i.e., a 5-tuple flow) and 2) the frequency distribution of all the flows. In addition, the overlay network maintains states for every flow of the subscribed customers, which may face scalability issues when an attack contains spoofed network traffic.

Xu et al. [213] and Zheng et al. [229] explore the viability of using OpenFlow switches to measure traffic statistics and detect attacks on service hosts. Both solutions utilize the limited OpenFlow rule space to perform threshold-based detection inline. SPHINX [43] analyzes the vulnerabilities with existing OpenFlow-based SDN networks and provides threshold-based techniques to detect attacks on SDN controllers.

SPIFFY [85] argues that attack traffic can be indistinguishable from legitimate traffic. Its defense approach is then to force attack traffic to become distinguishable. First, SPIFFY requires a transit network to temporarily increase the link bandwidth of a service. It then observes the traffic volume of each network flow towards the service. Next, the authors argue that adversaries will save their attack costs by fixing the traffic sending rate of each bot. Therefore, once the service's link bandwidth is increased, the network flows that do not increase their sending rate are attack flows.

3.2.4 DDoS Detection Summary.

In this section, we reviewed DDoS detection solutions by their deployment locations, and we summarized the quality of each solution in Table. 3. These detection solutions range from using static thresholds to machine learning techniques; the trend shows that researchers continue to improve the accuracy and robustness of their DDoS detection solutions throughout the years. While the existing DDoS detection research covers all common DDoS attacks at different network layers (e.g., protocol, application), we find many issues with existing DDoS detection research.

Table 3. Reviewed DDoS Detection Solutions

	Main Idea	Example	Performance	Scalability	Deployability
Source-end (Sec. 3.2.1)	TCP profiling	D-WARD [131]	Good	High	Low
Service-end (Sec. 3.2.2)	Static thresholds	Lakhina et al. [101] Soule et al. [191] Li et al. [106] Silveira et al. [181]	Poor	High	Medium
	Machine learning	Gavrilis et al. [61] Osanaiye et al. [150]	Fair	Medium	High
	Temporal analysis	Hussain et al. [72, 73] Sun et al. [196]	Good Good	Medium Medium	Medium Medium
		COSSACK [151]	Good	High	Low
		DDoS Shield [162, 163]	Good	Medium	Low
	Traffic entropy	Nychis et al. [146] Xie et al. [217] Xiang et al. [212] Behal et al. [15]	Good	Medium	High
	Str. pattern recog.	Afek et al. [2]	Poor	Medium	Medium
	Baiting	Khattab et al. [90]	Fair	High	High
In-network (Sec. 3.2.3)	Static thresholds	LADS [172]	Fair	High	High
	TCP profiling	Zhang et al. [224]	Good	Medium	High
	Temporal analysis	FireCol et al. [55]	Good	Medium	High
	Baiting	SPIFFY [85]	Good	High	Medium

First, the latest DDoS attacks (as reviewed in Sec. 2.2) remain as challenges to the existing detection solutions. In fact, at the time of this writing, we only find two detection solutions [196, 224] attempted to address the pulsing attacks. We also find that many detection solutions make strong assumptions about the attacks. For example, a solution may assume a DDoS attack concentrates all of its attack traffic on a single destination IP address or, a solution may attempt to extract the common attack string from packet payload while network traffic can be encrypted or spoofed. Second, the evaluation results of DDoS detection solutions are often difficult to compare or validate. For example, researchers do not use the same public datasets, and most network datasets do not capture the attacks accurately. In addition, many evaluate their detection solutions using traffic generators that do not react to the changes in networks; such an evaluation setup does not represent a real-world network environment where detection and mitigation systems need to work together.

Next, many detection solutions overlook the deployment issues in the real world. Depending on the operational context, many projects face challenges in defense automation. Specifically, a detection system is subject to the *base rate fallacy* where researchers only consider the accuracy ratio of their solutions but not the absolute number of detection alerts. In other words, a detection solution should not only report a false positive rate, but also the number of false positive alerts in a given deployment environment. Network operators cannot blindly trust a detection system to automate the DDoS mitigation for *all traffic.* Instead, they need to examine generated alerts to confirm the attacks, and consider the consequences of mitigation decisions. If the number of alerts is high, the network operators cannot process them manually in a timely manner.

Finally, a detection solution's capability is limited by the available network information. If a detection solution is designed to work with network telemetry protocols such as NetFlow or sFlow, it may not be able to detect all DDoS attacks. For example, NetFlow often aggregates packets into flows at minute-level, and sFlow typically samples packets. Meanwhile, if a detection solution is designed to work with network packets directly, it needs to consider its detection speed. If the detection algorithm does not scale, the value of delayed detection results is questionable. We see network measurement systems, such as defined in [220, 223], utilize low-latency memory SRAM and TCAM resources to measure network traffic more efficiently. For example, OpenSketch [223] utilizes an FPGA board to track the number of unique connections per host on high-speed network links. We believe future DDoS detection solutions should incorporate these systems as part of their designs.

3.3 DDoS Mitigation

The purpose of DDoS mitigation is to reduce the attack traffic that wastes a service's resources. An attacked service may start the mitigation locally (i.e., service-end mitigation) if the attack traffic does not deplete its network resource; the service can leverage the DDoS detection results to filter the attack traffic preemptively, which prevents the attack from wasting the service's host resource.

Unfortunately, DDoS attacks today can easily inundate the upstream network resource of services — a resource that is not controlled by the service. Therefore, attacked services often ask for mitigation efforts from the networks that carry victims' traffic. This simple ask is challenging to realize on the Internet. Because every AS on the Internet operates autonomously, it is not trivial to design a general mitigation system that works for all ASes. We should also realize that many ASes simply do not have the will to offer or invest in DDoS mitigation solutions. Because the local (service-end) mitigation techniques are similar to the service-end prevention techniques (i.e., Sec. 3.1.2), we will not repeat them in this section. Instead, we review **source-end** and **in-network** mitigation solutions for attacks that we cannot mitigate locally.

We divide in-network mitigation solutions based on their required level of system deployment rate. Specifically, we define an in-network mitigation system as *collaborative* when every transit network can mitigate attack traffic with the victim. Such a system requires a high deployment rate to perform effectively. On the other hand, an in-network mitigation system is *redirection-based* when it redirects a victim's traffic to a set of networks (often owned by one company) and then mitigates the attack. Such a system requires a low deployment rate.

Figure 11 shows two mitigation models that cover both the source-end and in-network mitigation solutions. Moreover, the figure lists the design factors of a mitigation solution. Namely, the mitigation solution needs to know the characteristics of the attack traffic. The DDoS detection results directly impact how good or bad the mitigation efficacy will be. Then, the solution needs to know where the attack traffic is on the Internet. If the solution blindly deploys traffic filters at arbitrary networks, it can impose collateral damages to a service or a network of services. Finally, it needs to demonstrate its mitigation capability and performance. For example, the solution cannot fully utilize perfect detection results if it can only filter traffic by destination IP addresses, or its runtime complexity leads to a high cost to mitigate large-scale attacks.

3.3.1 Source-end mitigation.

Source-end mitigation is used to remove DDoS traffic at edge networks that host DDoS bots. Due to the sheer number of edge networks on the Internet, such a solution requires a high deployment rate to account for different DDoS attacks; each attack may be sourced from an arbitrary set of compromised hosts. Some solutions, such as speak-up [207], even require the clients of services to participate in the defense. In general, a non-technical but critical challenge source-end mitigation solutions face is their deployment incentives. On the other hand, source-end mitigation is inherently more scalable than in-network mitigation.

D-WARD [131] requires system deployment at traffic source routers, e.g., a home router. The deployment location requires a router to track a limited set of flows without imposing significant performance penalties. D-WARD imposes rate limits on detected attack flows and monitors the attack flows continuously. Once the attack flows behave like benign flows, D-WARD lifts their rate limit on

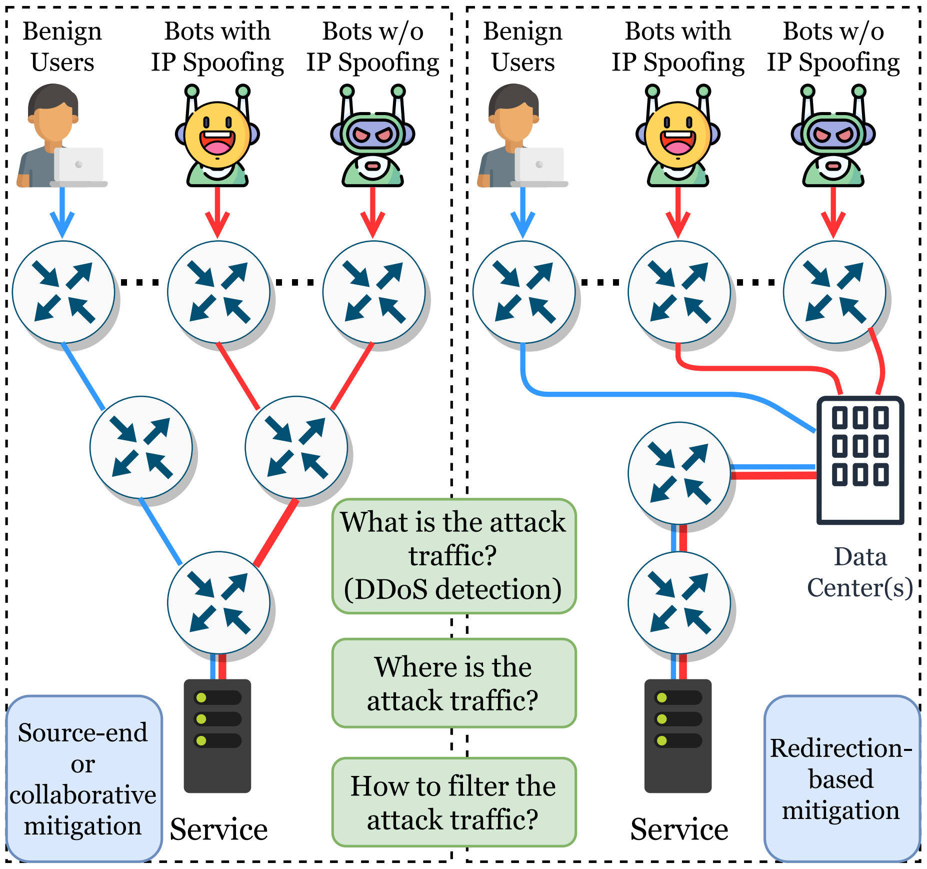

Benign Users
Bots with IP Spoofing
Bots w/o IP Spoofing
Benign Users
Bots with IP Spoofing
Bots w/o IP Spoofing
What is the attack traffic? (DDoS detection)
Where is the attack traffic?
How to filter the attack traffic?
Data Center(s)
Source-end or collaborative mitigation
Service
Service
Redirection-based mitigation

Figure 11. Design factors of a DDoS mitigation system

the flows. FR-WARD [126] is a similar source-end defense system that leverages the TCP flow control mechanism to rate limit abnormal flows. Specifically, an FR-WARD-enabled router fabricates three duplicate TCP ACKs and sends the ACKs to the potentially misbehaving host. After receiving three duplicate ACKs, FR-WARD assumes that benign hosts will reduce their traffic sending rate per TCP design; the bots will not reduce their traffic sending rate.

COSSACK [151] relies on IP traceback solutions to identify source-end networks that carry attack traffic. In COSSACK, the victim provides each source-end network the criteria to detect attacks. Each network then monitors for the corresponding attacks and mitigates any detected attack flows.

Huici et al. [71] proposed a mitigation system that requires edge networks to establish IP-to-IP tunnels with the victim network. Once the tunnels are established, the victim network can reject all traffic that is not coming from the tunnels. The victim network can also know where the attack traffic, spoofed or not, is coming from; each IP-to-IP tunnel maps to a source-end network. Once attack flows are detected, the victim can ask the corresponding networks to filter the attack flows.

Speak-up [207] takes a novel approach to mitigate DDoS attacks: the benign clients of a victim service send redundant requests to the service during attack time. Speak-up aims to allow benign clients to acquire fair shares of the link bandwidth towards the victim server. Speak-up requires software modification on hosts and servers, and in the case when a server is oversubscribed, speak-up *encourages* its clients to send redundant requests to compete with the malicious requests. In other words, if the attacker maintains a constant attack volume, benign clients can collaboratively send more traffic to reduce the goodput of the attack traffic.

3.3.2 Collaborative in-network mitigation. In a collaborative in-network mitigation system, multiple ASes collaborate with DDoS victims to mitigate DDoS attacks. In such a system, the number of ASes and whether the ASes carry attack traffic imposes an upper bound of the mitigation efficacy; the system cannot filter any DDoS traffic if none of the ASes carry the DDoS traffic. As a result, the mitigation cost per AS reduces as the system deployment rate increases.

AITF [9, 10] utilizes the filtering capability (e.g., ACL) on routers at different ASes. The authors realize that the low-latency memory (i.e., CAM/TCAM) space in a router is limited, which is remotely sufficient to filter large-scale DDoS attacks. They suggest that if each router from an AS can contribute a few thousand hardware filters, the transit ASes as a whole can support fine-grained DDoS mitigation with millions of filters. The AITF system starts DDoS mitigation on routers that are close to the source-end networks. The AITF-enabled routers implement an IP traceback [184] system that stamps their identifications to each packet. Thus, victims can know the responsible ASes and routers for carrying attack packets. AITF justifies its deployment incentive as follows: an ISP can either participate in the AITF system or lose its connectivity to the victim during a large-scale DDoS attack.

StopIt [108] recognizes that all filter-based mitigation systems are susceptible to filter exhaustion attacks; the attackers can spoof source IPs of the attack traffic to evade the filters. To address this issue, StopIt relies on Passport [107] to prevent IP spoofing. Unfortunately, Passport [107] does not provide a strong incentive for network communities to deploy.

Ballani et al. [13] proposed an *off-by-default* network where the system allows each end-host to specify whether it wants to receive traffic from the Internet. In other words, all routers in this system do not forward traffic to a host unless the host asks the routers to do so. Because each router in this system needs to maintain the state from each host, the scalability of this work is a concern.

DefCOM [147] consists of an attack detection component and an attack mitigation component. The former is deployed at the victim networks while the latter is deployed in the transit networks. The transit network routers impose rate limits on the malicious flows detected by the victim networks. E.g., a DefCOM router may allocate 90% of the link bandwidth to benign flows so that the attack flows share the remaining 10% link bandwidth.

SENSS [161] is a DDoS mitigation system that leverages software-defined networking (SDN). SENSS requires each participating AS to implement a set of application programming interfaces (APIs) that provide a DDoS victim to 1) query traffic statistics from the SENSS-enabled ASes and 2) control if SENSS-enabled ASes should forward packets en route to the victim. The work suggests victim networks may detect Crossfire attacks [86] by collecting traffic statistics from their upstream networks. Specifically, a victim network can ask its upstream networks whether they see any link congestions. Then, SENSS-enabled networks may run attack detection to detect Crossfire attacks. However, the challenge is that upstream networks may not know the normal network condition of its customers.

Dietzel et al. proposed Stellar [44], a mitigation system that leverages existing RTBH infrastructure to disseminate fine-grained filters to major Internet Exchange Points (IXPs) to address the deployment issues that the research above faces. Each fine-grained filter is encapsulated in a BGP message. As more edge networks peer with IXPs, authors believe IXPs have a unique advantage in mitigating DDoS attacks on the Internet.

While most in-network defense systems focus on inter-AS collaboration, dFence [115] is an intra-AS DDoS defense system to enable ISPs to provide DDoS mitigation services to their customers within its capability. The authors of dFence realized that many proposed DDoS systems require either software modification at either routers, hosts, or both. dFence dynamically redirects a service's traffic to an ISP's middleboxes via intra-domain routing protocols. The middleboxes offer both stateless and stateful traffic mitigation (i.e., check for valid TCP connections). The work employs consistent hashing on network connections to ensure that the same middlebox will process each network connection bi-directionally.

In CoDef [103], a victim directly sends traffic control requests to each source-end AS. For example, if a source-end AS contains only the benign traffic, the victim requests the AS to forward traffic to the next-hop AS whose link is not congested. However, if the AS contains the attack traffic, the victim asks the AS to reduce its traffic en route to the victim. In other words, CoDef's traffic mitigation granularity is at AS-level.

3.3.3 Redirection-based in-network mitigation.

Mitigation systems in this category rely on DNS or BGP to redirect the traffic towards a victim service through the networks (e.g.data centers) to filter attack traffic (as illustrated in Figures 9 and 11). Once the traffic (malicious or benign) traverses through networks equipped with ample network and computational resources, these systems then balance the traffic within each network, cleanse the traffic, and forward the cleansed traffic to the victim service. The process is widely adopted by DPS providers today.

However, the Internet is an unicast network. We can only mimic an anycast network with the help of DNS or BGP. The mimicked anycast network does not fundamentally determine the traffic forwarding decisions for other end hosts or autonomous networks. For example, a mimicked anycast network may attract too much traffic for a data center to handle. Indeed, Moura et al. [135] and Koch et al. [93] point out that further evaluation on the existing anycast network is required to understand its traffic redirection efficacy.

CenterTrack [194] is an early work that leverages BGP announcements to redirect a service's traffic to an overlay network. The overlay network tracks the traffic forwarding path of each packet en route to the service and has the flexibility to choose where to filter attack traffic. The overlay network also handles bi-directional traffic from and to the service. Such a requirement allows the CenterTrack system to determine if a client is sending traffic that is violating the design of a protocol (e.g., whether the TCP sequence numbers are aligned between the client and the server). The system requires the service to either (1) connect a physical link to the overlay network or (2) establish a network tunnel (e.g., GRE) with a node in the overlay network to send/receive its traffic.

MiddlePolice [109] leverages both DNS and BGP protocols to redirect a service's traffic through data centers. Once packets arrive in a data center, each MiddlePolice node (mbox) mitigates the attack packets with rate limiting and packet filtering techniques. Meanwhile, the mboxes mark packets to indicate that the packets are processed by the MiddlePolice system. If any packets are not affected by its traffic redirection, i.e., the packets are not processed by any mbox, MiddlePolice requires transit networks to drop unmarked packets. However, the traffic redirection technique is not perfect, and benign traffic also may not go through any mboxes, which means the transit networks may also drop benign traffic.

Bohatei [50] proposes a mitigation system for ISPs to offer DDoS mitigation as a service to their customers. During an attack, a Bohatei ISP first distributes its customer's traffic to geographically distributed data centers. The ISP then measures the traffic volume towards the victim and estimates the number of hosts required to process the traffic. Bohatei uses IP tunnels to funnel traffic to multiple data centers, where each data center spawns mitigation processes on virtual machines and balances attack traffic load to the machines. The work assumes that

Table 4. Reviewed DDoS Mitigation Solutions

	Main Idea	**Example**	**Performance**	**Scalability**	**Deployability**
Service-end	See Table. 2	N/A	N/A	N/A	N/A
Source-end (Sec. 3.3.1)	Bandwidth competition	Speak-up [207]	Fair	Medium	Low
	Rate limiting	CoDef [103]	Fair	High	Med
		D-WARD [131] COSSACK [151]	Good	High	Medium
	Whitelisting	Huici et al. [71]	Good	Low	Medium
Collaborative (Sec. 3.3.2)	Whitelisting	Ballani et al. [13]	Poor	Low	Low
	Software filtering	dFence [115]	Good	Medium	Medium
	Rate limiting	DefCOM [147]	Good	Medium	Medium
	Blacklisting	Jin et al. [79]	Poor	High	Low
		ATIF [9, 10] StopIt [108] SENSS [161]	Good	High	Medium
		Stellar [44]	Good	High	High
Redirection (Sec. 3.3.3)	Routing changes	RAC [186]	Poor	Medium	Medium
	Overlay network	CenterTrack [194]	Good	Low	Medium
	Software filtering	MiddlePolice [109] Bohatei [50]	Excellent	Medium	Medium

the ISP's ingress links are not congested, or it is too late to distribute traffic among the data centers.

As many criticize the practicality of DDoS mitigation solutions, Smith et al. [186] proposed *routing around congestion (RAC)*, a BGP-based solution to mitigate link-flooding attacks against transit networks (e.g., the Crossfire attack [86]). The solution relies on carefully crafted BGP announcements. Specifically, a victim network sends BGP messages to its upstream ASes and hopes to force the ASes to divert their traffic to a healthy link. RAC assumes that the victim knows: (1) link-flooding attacks such as defined in [86] are occurring, (2) congested links on the Internet, and (3) there exists alternative links to distribute the attack load to. However, later work from Tran et al. [200] concludes that RAC is unattainable in the real world.

3.3.4 DDoS mitigation summary. We summarized the mitigation solutions above in Table 4. We rank the performance of a mitigation solution as good or excellent when it meets its mitigation goal and has no strong assumptions, and we assign a poor or fair score, vice versa. Many mitigation systems filter or rate limit attack traffic at a fine granularity. Recent work such as those in [50, 109] proposed systems to filter traffic in software with arbitrary matching patterns. Therefore, the research community has DDoS mitigation systems that can take advantage of fine-grained DDoS detection results.

The deployment location of a mitigation solution directly affects its scalability. Source-end mitigation solutions scale the best as each deployment only processes an edge network's traffic. Many reviewed collaborative-based mitigation solutions (such as those defined in [10, 108, 161, 44]) consider scalability as an important factor in their design. Therefore, their scalability is high. However, existing redirection-based mitigation solutions can face scalability issues. Specifically, because such a solution is often deployed by a single AS, the solution is more difficult to scale to protect multiple networks simultaneously.

We rank most mitigation solutions at a medium level of deployability. While they are technically viable to deploy, they do not provide strong incentives for ASes to deploy. Moreover, their implementation difficulty and maintenance cost is not trivial. We assign Stellar [44] a high deployability score because it leverages existing traffic filtering infrastructures such as RTBH and BGP FlowSpec.

Apart from the system research above, there exist mitigation solutions that focus on more specific performance issues. For example, Soldo et al. [189, 188] study how to generate a fixed number of traffic filters to mitigate attack traffic with little collateral damage. Armbruster et al. [12] study the minimum set of nodes to deploy ingress/egress filtering to prevent IP spoofing on the Internet. Zhang et al. [226] and Sisodia et al. [185] model different filter placement strategies to study their efficacy when deploying fine-grained DDoS filters in transit networks. You et al. [222] proposed an auction-based DDoS filtering model to achieve reduced

DDoS mitigation costs. These solutions complement the above mitigation systems in more specific tasks.

CHAPTER IV
ON CAPTURING DDOS TRAFFIC FOOTPRINTS ON THE INTERNET – TOWARD BETTER DDOS DEFENSE INPUT

This chapter presents an AS-level DDoS traffic footprint capturing system that can serve a DDoS defense system with better input and increase the efficacy of DDoS defense. Specifically, the system can inform a DDoS defense system what autonomous systems (AS) carry traffic from particular source addresses and the amount of traffic each AS carries in a near real-time stream. We quantify the benefit of the traffic footprint capturing system under multiple DDoS attack scenarios to show how critical the input is to DDoS defense systems. Furthermore, we evaluate the system's core data structure's speed and measure the system cost under different system operational models. We therefore introduce the PathFinder system as a service for DDoS defense systems.

The content in this chapter appeared in [180, 177]. Lumin Shi is the leading author of this work and responsible for leading all the presented analyses.

4.1 Introduction

Today's Internet is vulnerable to distributed denial-of-service (DDoS) attacks. During a DDoS attack, an attacker controls many compromised machines to send unwanted traffic toward the victim in order to exhaust the network or computational resources needed to access the victim. DDoS attacks have become more frequent and damaging to many network services [5]. An attacker with ample resources can easily launch DDoS attacks with an overwhelming traffic volume and take down nearly any Internet service. For instance, a large-scale DDoS attack on Dyn [221] easily disabled its domain name service and crippled many major web services that relied on it such as Twitter, PayPal, and over fifty others for hours. DDoS attacks have also become more sophisticated, such as those described in [86, 88, 179].

While many DDoS defense systems have been proposed, a primary challenge in effectively defending against DDoS attacks is that a DDoS defense system usually has little knowledge regarding which paths DDoS traffic has traveled along, how much traffic traveled along each path, and also which source addresses or prefixes of the DDoS traffic are associated with each path. Such information about the DDoS traffic, which we collectively call **DDoS traffic footprints**, if available, can enable a DDoS defense system to become more informed and thus more effective in handling DDoS attacks. For example, it may learn which autonomous systems (ASes) or AS paths have seen a large amount of traffic to the victim, thus more preferably deploying DDoS traffic filters there; it can also know better which source addresses (or more likely the source IP prefixes, for scalability) to filter in the case of source-based filtering; and it could also conduct traffic pattern analysis if the footprints are continuously provided. Fig. 1 shows an example: while the DDoS traffic toward the victim originates from—and is forwarded by—many ASes, if the DDoS defense system knows the AS paths of the DDoS traffic toward the victim, it can request that specific ASes on the paths, such as AS 2 and AS 6, filter the DDoS traffic.

In fact, even when facing more complex DDoS scenarios, a DDoS defense system can still benefit from DDoS traffic footprints. First, when a DDoS attack employs large-scale IP spoofing, as long as the defense can determine which traffic is DDoS traffic, with the DDoS traffic footprints, such as the paths of DDoS traffic or which ASes carry the traffic, the defense will still know correctly where to deploy DDoS traffic filters *en route* and mitigate the DDoS attack effectively. On the other

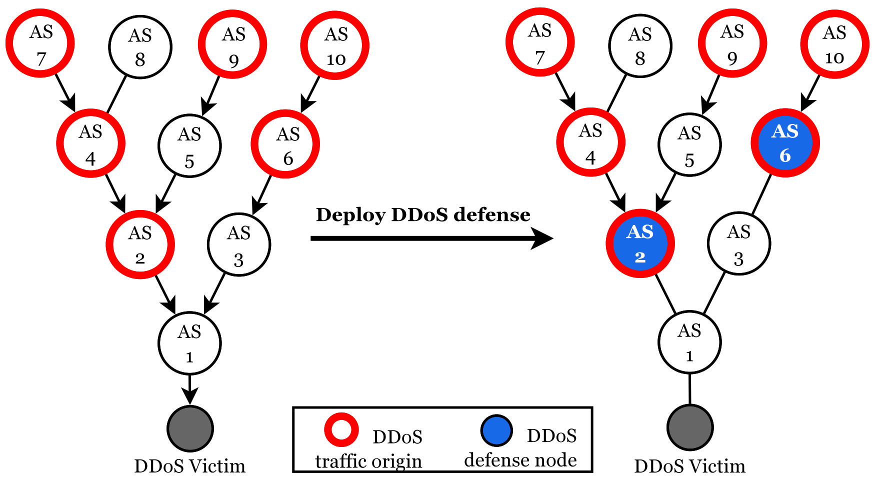


Figure 12. A DDoS attack and defense example.

hand, the traffic footprints can help detect IP spoofing if they show that traffic from the same IP addresses to the DDoS victim originated from different ASes. (Here, we leave out the details on determining whether given traffic is DDoS or spoofed (or both), which is the responsibility of DDoS defense itself and out of the scope of this chapter.) A potentially concerning issue is the dynamic nature of DDoS traffic paths since a DDoS attack may periodically switch to a new batch of DDoS bots to launch its attack, likely from different paths [208, 209]. Here, we can safely assume the resource for any DDoS attack is limited, including the number of DDoS bots as well as the paths incurred by the DDoS traffic. So, while it is possible for a DDoS attack to change the bots and shift the attack paths, it will eventually exhaust all potential bots and possible attack paths, and a DDoS defense system can still benefit if it learned the IP sources of bots and their paths used to attack a DDoS victim. Plus, as the DDoS traffic approaches the victim, even if the DDoS traffic keeps changing their paths, the paths will converge, allowing the PathFinder to discover their common hops where DDoS filters may be deployed. Finally, some DDoS attacks are transient; however, as long as they last longer than what it takes to learn and report their footprints, a DDoS defense solution can still leverage the footprints in mitigating such DDoS attacks.

Various approaches to obtaining such information could be considered, including numerous IP traceback approaches and path inference methods that aim to address the asymmetric nature of Internet paths and ascertain the paths traveled by DDoS packets to reach the victim. Unfortunately, as we will discuss in more detail in Sec. 4.2, these approaches have serious drawbacks. For example, the path inference methods are often inaccurate (e.g., the research in [122] claims an accuracy between 70% and 88%); IP traceback approaches [184] introduce significant changes to router hardware or software, rely on inter-AS collaboration, and need routers on the Internet to *constantly* monitor the traffic. These approaches are also not well-equipped to provide other footprint information, such as total or per-source bandwidth consumption information or the IP addresses or prefixes of DDoS sources.

We therefore introduce the **PathFinder** system as a service for DDoS defense systems. Upon request from a DDoS defense system on behalf of a DDoS victim, PathFinder can gather and provide the footprints of the traffic to the victim. We make the following contributions:

1. PathFinder consists of an architecture that is easy to implement and deploy on today's Internet. Every AS can join PathFinder without reliance on other ASes, and it employs an *on demand* service model with low overhead. PathFinder does not require *every* AS to participate in order for PathFinder to benefit DDoS defense.

2. We design the setup and operations of each component of the architecture while considering a series of real-world factors and the high speed and large scale of DDoS traffic.
3. We design a new data structure called **PFTrie** that supports fast and easy storage and retrieval of traffic footprint information, with a set of PFTrie optimization methods.
4. We design a streaming mechanism that manages the timing of transmitting traffic footprints, including adaptively adjusting the transmission interval to lower the overhead.
5. We also introduce a new mechanism called *zooming* that enables PathFinder to dynamically adjust its granularity in collecting DDoS footprints for better efficiency.
6. We evaluate PathFinder and show that PathFinder can significantly improve the efficacy and efficiency of DDoS defense, its PFTrie is fast and has a manageable overhead, its streaming and zooming mechanisms are fast and efficient, and PathFinder can also sample DDoS traffic at certain rate over a time window to still capture DDoS footprints with high accuracy.

The rest of this chapter is organized as follows. We first discuss related work in Sec. 4.2, followed by an overview of PathFinder in Sec. 4.3. We then describe individual components of PathFinder, including the PathFinder monitor in Sec. 4.4, the PFTrie data structure for traffic logging in Sec. 4.5, the PathFinder proxy in Sec. 4.6, the streaming mechanism in Sec. 4.7, and the zooming mechanism in Sec. 4.8. We evaluate PathFinder in Sec. 4.9, discuss several open issues in Sec. 4.10, and conclude the chapter in Sec. 4.11.

4.2 Related Work

4.2.1 IP Traceback. IP traceback, first introduced in [21], allows a victim to trace the source of an IP packet it has received and reconstruct the router-level path taken by the packet, even if the source address of the packet is spoofed. While many IP traceback solutions have been proposed [184], marking and logging are the two most well-developed approaches.

In a marking approach, such as those described in [169, 215, 11, 3, 144, 136, 154], when a router along a path forwards a packet to the victim, the router marks the packet with its own IP address (or its hashed result) or an edge that the packet has traversed, typically using some unused fields in the IP header of the packet. When the victim receives *enough* marked packets, even if routers *en route* mark packets with certain probability rather than all the time, the victim can then reconstruct the paths of these packets (assuming the paths are stable).

In a logging approach, such as those described in [187, 105, 68], as a router along a path forwards a packet to the victim, instead of marking the packet, the router uses some data structure (e.g., a Bloom filter [18]) to store the digest of the packet (rather than the packet itself in order to save space), enabling it to later determine whether it has seen the packet. When the victim wants to trace a packet, it can query its upstream routers, asking whether they have seen the packet. Similarly, a router that has seen the packet can query its neighboring routers about the packet, and so on. Eventually, the victim can reconstruct the packet's path using an ordered list of routers that have seen the packet.

PathFinder has advantages over existing IP traceback approaches in the following respects:

1. *Operation*: PathFinder is also essentially a logging approach, but while existing IP traceback approaches record packet information or mark packets constantly, PathFinder is an on-demand service and PathFinder monitors will only record traffic information when requested.
2. *Overhead*: Because it operates on demand, PathFinder incurs much less operational overhead than existing IP traceback approaches. In addition, packet marking approaches will modify packets before forwarding them, which will introduce delays in processing packets and could downgrade the network throughput significantly, especially when dealing with a high-bandwidth link.
3. *Accuracy*: The accuracy of the marking approaches depends on how many marked packets the victim can receive to reconstruct the paths of packets. The accuracy can suffer if the victim cannot receive enough marked packets, such as when its inbound link is congested with DDoS traffic. The existing logging approaches as well as PathFinder, on the other hand, so long as their monitoring mechanism can process packet headers at line speed, can log packet information with little loss and reach a high accuracy.
4. *Deployability*: Existing IP traceback approaches face obstacles for deployment: Whether based on marking or logging techniques, they introduce significant hardware or software changes to routers and also require inter-AS collaboration. Further, as marking approaches overload existing IP header fields with marks, they may interfere with the defined functionality of those fields and even cause confusion at network equipments that are not aware about marking functionalities. PathFinder instead introduces few changes to routers, and PathFinder-participating ASes talk directly with a PathFinder proxy and do not need to communicate with each other.

4.2.2 Path Inference.

Researchers have studied how to infer the path between two end points on the Internet. Without assuming any control over the network infrastructure or access to end points, research in [122] investigated how to leverage Border Gateway Protocol (BGP) tables collected from multiple vantage points to infer the AS path between any two end points on the Internet. Also, via the probing from multiple vantage points and the IP timestamp and record route options, research in [87] proposed a "reverse traceroute" to allow a user to infer the path from a remote end point to the user, without accessing the remote end point. Inference-based approaches do not require changes to network equipment and are easy to deploy. However, they are generally subject to some degree of inaccuracy (e.g.the accuracy from the research in [122] is between 70% and 88%). Moreover, since they are not based on watching traffic in real time, they will not be able to report the bandwidth consumption and other traffic-related information associated with the inferred path. Similarly, path inference approaches are not suited for finding the paths of DDoS traffic using spoofed IP addresses, as they require the true IP addresses of DDoS bots to infer the paths of DDoS traffic.

4.3 PathFinder Overview

4.3.1 PathFinder as a Service for DDoS Defense.

We designed PathFinder as a service for DDoS defense. Upon request from a DDoS defense system, PathFinder can provide the footprints of all the traffic toward a victim that each participating AS has witnessed. Note that PathFinder does not distinguish DDoS traffic from the legitimate traffic, which PathFinder assumes to be the job of the DDoS defense. The footprints include:

- all the AS paths taken by the traffic;
- if requested, the source IP addresses or prefixes of the traffic; *and*

– if requested, the amount of traffic per source address, or per source prefix, or per AS *en route*, or other information about the traffic.

Note that DDoS defense systems typically inform ASes, not routers or mitigation appliances in those ASes, which traffic should be filtered. It is up to an AS to determine internal locations of filtering DDoS traffic. PathFinder therefore only needs to record AS-level paths of DDoS attacks.

When requesting the PathFinder service, a DDoS defense system can specify to PathFinder a set of parameters regarding the traffic footprints, including:

– the destination address of the victim, which could be an IP address or prefix, or an IP address plus a port number;

– the length of time for which to collect the traffic footprints (typically during the DDoS attack);

– from which ASes (if not all the ASes supporting PathFinder) to collect the traffic footprints;

– whether to collect the source addresses or prefixes of the traffic, and if so, the prefix granularity (e.g./24 to learn all the /24 prefixes; /32 to learn all the /32 prefixes, i.e., all the source IP addresses); *and*

– whether to collect the bandwidth consumption of the traffic, and if so, the granularity (per source address, per source prefix, or per AS *en route*) and the unit (packets per second or bits per second).

4.3.2 Architecture. PathFinder is a log-based system that enables a client—which is any DDoS defense system in this chapter—to learn the AS paths, sources, bandwidth consumption, and other information concerning the traffic

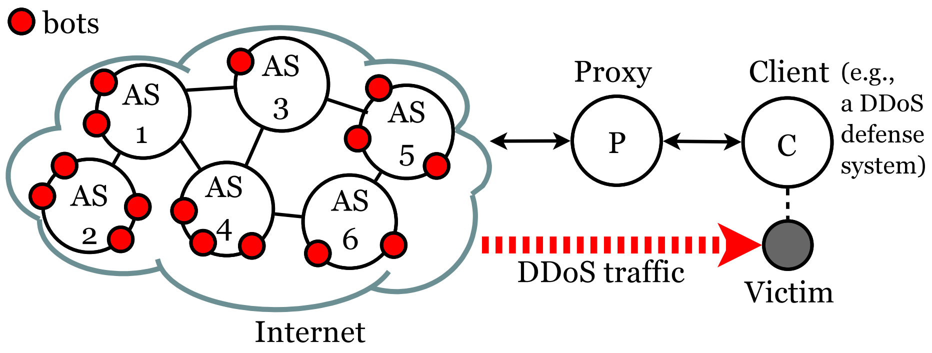


Figure 13. PathFinder architecture.

toward a DDoS victim, i.e., the footprints of the traffic. As we will show in the rest of this chapter, PathFinder is easy to deploy as it requires minimal reconfiguration of routers; it is scalable as it will continue to perform well if there is more traffic from more sources or if more ASes support PathFinder; and it is accurate, fast, and efficient in providing the traffic information.

Fig. 13 shows the high-level architecture of PathFinder. It consists of three types of entities:

1. **PathFinder clients** which interact with their proxy to request the PathFinder service and retrieve from their proxy the footprints of the traffic to a DDoS victim;
2. **PathFinder proxies** which (1) pass their clients' requests (including all parameters described in Sec. 4.3.1) to the PathFinder monitors at all participating ASes; (2) receive and process PathFinder logs (**PFLogs**) from these monitors, which they use to derive the DDoS traffic footprints; and (3) return the footprints to their clients; *and*
3. **PathFinder monitors** at all participating AS which, according to the request from a proxy, (1) process the traffic that their AS originates or forwards towards the client specified in the request; (2) generate PFLogs of

the traffic, which record the AS path, source addresses (if requested), and amount (if requested) of the traffic; and (3) return the PFLogs to the proxy. Note that monitors from different ASes do not need to interact with each other, thus avoiding any reliance on inter-AS collaboration.

4.4 PathFinder Monitor

4.4.1 Addressing Design Requirements. The monitor at each PathFinder-participating AS faces two design requirements. First, the monitor must consult routers within the AS to learn the AS path from the AS to the victim. Second, it must access the traffic toward the victim in order to record their source addresses and/or amounts, if requested. As an AS can have a complicated topology with inter-connected border routers and internal routers, some routers may not be on any path toward the victim at all and some may be on the same path. To meet both requirements, for every path of the traffic to the victim, the monitor must be able to talk with at least one router on that path in order to learn its AS path to the victim and to access the traffic it forwards to the victim. For the former (i.e., to learn the AS path), as every border router of the AS runs BGP and maintains a Routing Information Base (RIB), the monitor can query the RIB at a border router of the AS. Note that BGP implementations such as Cisco IOS [33] and FRRouting [57] all support such a query. For the latter (i.e., to access the traffic), the monitor must apply traffic mirroring or tapping techniques (we rule out possible hardware telemetry support from routers; although they produce traffic records in formats like NetFlow or IPFIX, the records are only exported at a fixed interval, often with a long delay).

The monitor may further face a third requirement if it needs to produce PFLogs to record traffic sources as well as source-based information such as traffic amounts per source. There could be a huge amount of traffic from many distinct sources toward the victim, especially if the victim is currently under a severe DDoS attack, thus making it challenging for the monitor to record all the sources and their corresponding bandwidth consumption at high speed. The most obvious solution is to use a digest-oriented data structure such as a Bloom filter or hash table. However, while the monitor can use a Bloom filter to easily answer whether it has seen an IP address or prefix or not, it is not good at listing which specific source IP addresses or prefixes it has seen. A hash table is better, but it is not flexible in processing or aggregating IP address and prefix information, thus not scalable when the logs are of a huge size. We therefore design a new, trie-based data structure called **PFTrie** to facilitate the recording and transmission of PFLogs, which we detail in Sec. 4.5.

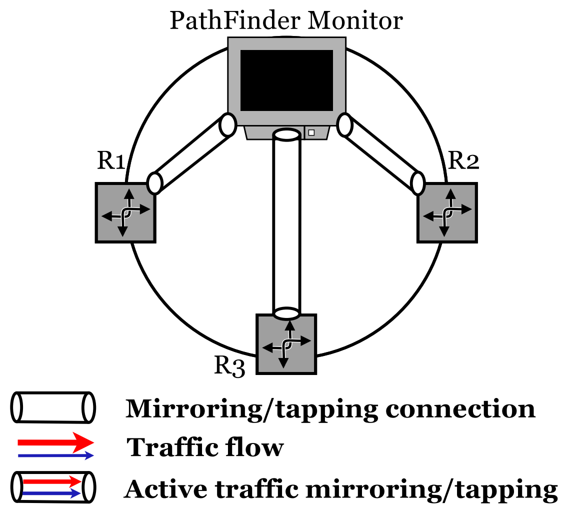
4.4.2 Setup. A PathFinder-participating AS must set up its PathFinder monitor and its working environment. First, the monitor needs to arrange traffic mirroring or tapping with every border router in order to obtain their traffic *in real time* when needed. To do so, given the autonomy of ASes, each AS may adopt its own preferred procedure. For a small AS without many routers, it can physically wire the monitor with every router for wiretapping (Fig. 14a shows an example). For a large AS with many routers over a large geographic region, it can use virtual circuits for traffic mirroring with the routers [35, 81]. Further, a large AS may employ multiple monitors, with one monitor configured as a master monitor that can assign the workload across all the monitors. (Without losing generality, we assume one monitor per AS in the rest of the chapter.)

Second, the monitor must be able to remotely login to each router that it is wired with and execute commands on that router, such as querying the AS path or the next hop from the router to any destination IP address. To do so, we assume every router supports secure shell (*ssh*), which is true for most routers nowadays [32, 82].

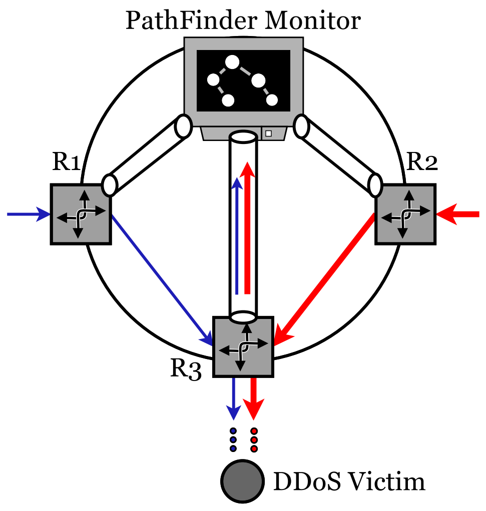
Finally, the monitor must be easily discoverable by every PathFinder proxy. While the monitor can employ a running daemon process with a publicly known port number, proxies must also know the monitor's IP address. Many options exist; for example, the monitor can maintain its IP address and other information at a web page. Or, the AS can set up a Domain Name System (DNS) record for its PathFinder monitor; e.g.`3582.pathfinder.org` may point to the PathFinder monitor of `AS 3582`. The AS can also, at its discretion, use standard access control mechanisms to limit which remote parties can query its PathFinder monitor.

4.4.3 Operation. The monitor at every PathFinder-participating AS operates on demand, remaining idle unless it receives a request from a PathFinder proxy, in which case the monitor will learn the IP address or prefix of the victim in question, together with the parameters in the request as defined in Sec. 4.3.1, and start to generate PFLogs on behalf of the victim.

The first step that the monitor takes is to identify a set of routers in its AS that are both necessary and sufficient to capture all the possible traffic that the AS may originate or forward toward the victim. Note the AS may also originate traffic toward the victim from its own routers. We therefore use all possible egress routers from which the traffic to the victim may exit the AS. For example, in Fig. 14b, as the AS forwards two traffic flows toward the victim and both exit the AS from egress router $R3$, the monitor will select $R3$ to produce PFLogs for the victim. Further, it is straightforward to decide which routers are possible egress routers for the traffic to the victim. The monitor can query every border router's RIB to learn


PathFinder Monitor
R1
R2
R3
Mirroring/tapping connection
Traffic flow
Active traffic mirroring/tapping

(a) The setup of a PathFinder-participating AS.



(b) The snapshot when the monitor is capturing traffic.

Figure 14. An example setup of a PathFinder-participating AS.

its next hop to reach the victim's IP. If the next hop is a router still within the AS, the border router in question is not an egress router; otherwise, it is.

Once the routers are selected, the monitor then talks with them to collect and produce PFLogs. First, the monitor will query each selected router to retrieve its AS path to the victim from its RIB. Furthermore, if the client requests the bandwidth consumption information of the total traffic to the victim via the AS, since the monitor is mirroring or tapping the traffic from these routers (among others), the monitor can observe the traffic from these routers and count their total volume (# of packets or # of bits), either per time unit or over a period of time (as specified in the request or based on a default value). Note that only the traffic to the victim is mirrored or tapped and, more importantly, this traffic does not share the path of the production traffic, so it will not interfere with the production traffic.

If the client did not request the source addresses or prefixes of the traffic, or source-based bandwidth consumption or other information, PathFinder operates in **source-agnostic mode**. In this mode, the procedure already outlined above has collected all the PFLogs that the client requires. Otherwise, the monitor will operate in the **source-aware mode**, which requires gathering further information to produce source-based PFLogs, using the PFTrie data structure (see Sec. 4.5). For both modes, the monitor employs a **streaming** mechanism (see Sec. 4.7) to deliver the PFLogs to the proxy of the client.

Finally, note that each monitor only communicates with PathFinder proxies. No inter-AS collaboration is needed. Monitors from different ASes are not required to communicate or collaborate, and each AS independently participates in PathFinder without any reliance on other ASes.

4.5 PFTrie—A PathFinder Data Structure for Traffic Logging

When in source-aware mode, the monitor at every PathFinder-participating AS will need to record all the sources that the AS has seen sending traffic to the victim in question, and if requested, the bandwidth consumption information per source. In doing so, the monitor needs to employ a data structure and accompanying algorithms to log sources, count bandwidth consumption, and transmit such data, all at a high speed to keep up with the line-speed packet arrival rate. Due to its speed, the trie data structure has been popular in storing IP addresses and prefixes for other purposes, such as those in the Forwarding Information Base (FIB) of software routers. A trie is also called a *prefix tree*, where every node on the trie uses its position on the tree to store the key of the node, such as the IP address or prefix represented by the node. We therefore adopt the trie data structure for this purpose. Furthermore, to meet the design requirements discussed in Sec. 4.4.1, we enhance the trie data structure and design as follows the PFTrie—a PathFinder data structure for logging traffic.

4.5.1 Basic PFTrie Operations.

The monitor captures and processes every packet toward the victim. It stores the IP source address of the packet into the PFTrie via a **put** process. In this process, the monitor may modify the PFTrie by adding new nodes to store the IP address, or it may discover that it has already created those nodes due to a previous packet with the same IP address. In either case, the put process will return a node representing the IP address, which the monitor can further update with bandwidth consumption information incurred by the current packet, if requested.

In the put process, the monitor will traverse the trie from the root downwards. It will traverse a node at each level—which we also call an **anchor**—to further move to the next level; clearly, when the traversal starts, the anchor is the root of the trie. At the same time, it iterates through the bits of the IP address, starting from the leftmost bit, as follows:

(1) If the current bit in the IP address is `0`, it traverses to the left child of the anchor, otherwise it traverses to the right child; either way, the chosen child becomes the new anchor.

(2) If the new anchor does not exist, the monitor will detect a fault, i.e., the trie has not stored this IP address yet; the monitor will then add the missing child onto the trie, and use this child as the new anchor. (Note that this new anchor will also avoid the same fault for the next time.)

(3) If the current bit is already the rightmost bit of the IP address, the monitor then knows that the IP address is stored in the trie as represented by the new anchor, and return the new anchor node. (Note the returned node is always

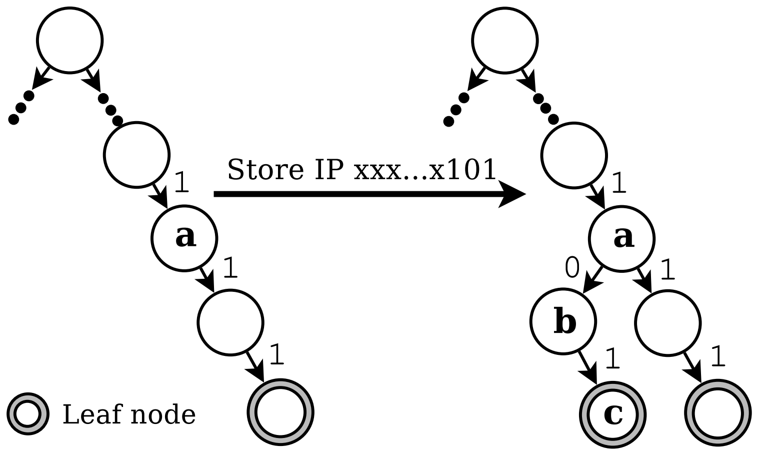


Figure 15. Store an IP address that ends with 101.

a leaf node on the trie.) Otherwise, it still needs to move to the next bit of the IP address; it then uses the new anchor as the current anchor to repeat step (1) above.

Fig. 15 shows an example of inserting a new IP address that ends with `101`. When it traverses to node a by following bit `1`, it needs to follow bit `0` to go to a's left child; since it does not exist, the monitor adds node b as a's left child. It then needs to follow the last bit `1` to go to b's right child, for which it adds a new node c. Now that the entire address is processed, the process returns node c representing the newly stored IP address.

4.5.2 PFTrie Optimization. We further optimize the PFTrie for a faster traversal process. First, with the design in Sec. 4.5.1, for every bit of an IP address, the trie must maintain a corresponding node at each level; for example, an IPv4 address will lead to 32 nodes at 32 respective levels on the PFTrie. We address this issue with two independent optimization methods: the bottom-up aggregation of leaf nodes 4.5.2.1 and the top-down collapse of prefixes 4.5.2.2. Furthermore, we introduce a method to avoid duplicate traversal of the PFTrie when a source address is already stored 4.5.2.3. We describe each method below.

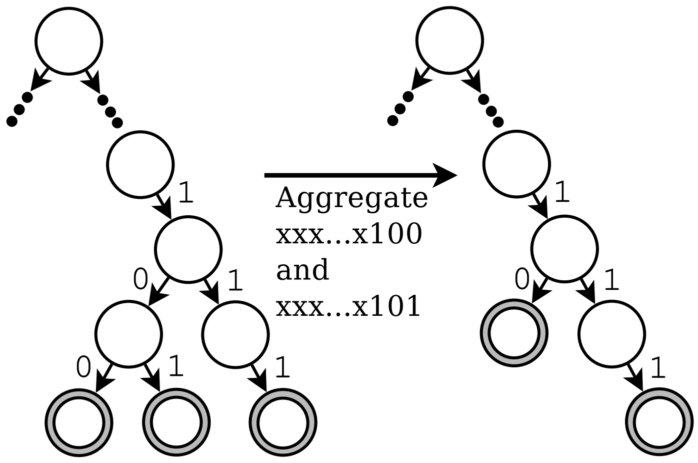


Figure 16. PFTrie optimization: Aggregating sibling leaf nodes into one new leaf node.

4.5.2.1 Bottom-up Aggregation of Leaf Nodes. Because of traffic locality, sometimes there can be multiple sources from the same prefix sending traffic to the victim. For example, besides seeing the 32-bit source IP `xxx...x101` to the victim, the monitor may also see traffic to the victim from another source IP `xxx...x100`, which only differs from the former source IP by the very last bit; in other words, they share the same 31-bit prefix. When such locality is detected, two leaf nodes at level 32 are not needed to represent the two IP addresses. Instead, as shown in Fig. 16, we can aggregate the two leaf nodes into a new, level-31 *leaf* node, indicating the monitor has seen traffic from both IP addresses in the 31-bit prefix. Furthermore, this aggregation can continue if the new leaf node has a sibling leaf node. Clearly, this bottom-up aggregation process can reduce the depth of certain branches of the PFTrie, thus speeding up the *put* process.

One challenge here is the logging of the bandwidth consumption information. After aggregation, the monitor could simply copy the bandwidth consumption of each old leaf node into the new leaf node; or, it can sum the bandwidth consumption of the two leaf nodes, recording the bandwidth

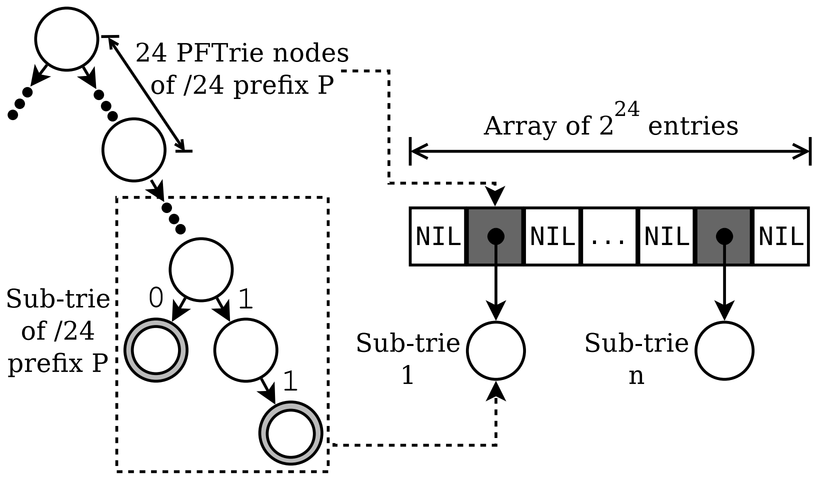


Figure 17. PFTrie optimization: Collapsing all /24 prefixes into an array with 2^{24} entries.

consumption of the IP prefix represented by the new leaf node. The choice here depends on the client's request regarding the prefix granularity for recording the bandwidth consumption information (e.g.a /32 prefix granularity means to record the information per IP address, while a /0 prefix means the total bandwidth consumption for the whole IP space).

4.5.2.2 Top-down Collapse of Prefixes. During a *put* process the nodes at the top portion of the PFTrie are frequently traversed. Rather than traversing these nodes one by one each time, we collapse them into all the IP prefixes they represent, allowing the *sub-trie* below each prefix to be reached by directly indexing an array. Fig. 17 shows an example of collapsing /24 prefixes into an array (the monitor can collapse prefixes of other lengths similarly, such as all the /16 prefixes). We populate the array with all /24 prefixes that exist in the PFTrie. For every /24 prefix, the monitor treats the 24 bits of the prefix as an integer, use the integer as the index to directly locate the entry of the array, and have that entry point to the sub-trie originally below the prefix. So, instead of traversing 24 nodes of a /24 prefix and then traversing the sub-trie of the prefix, the monitor can immediately locate the entry for this prefix in the array, access from the entry the sub-trie of this prefix, and then traverse the sub-trie as before.

4.5.2.3 Avoidance of Duplicate Traversal. So far, if the PFTrie has recorded an IP address, when a packet with the same IP address arrives, the monitor will still run the *put* process and traverse the PFTrie, only to find the IP address is already stored. With n packets from the same IP address, the overhead will multiply for n times.

We introduce a bitmap for each one of M most recently visited sub-tries. After storing an IP address in a sub-trie, the monitor will also set the bit in the bitmap corresponding to this IP to `1`, so that the *put* process for the same IP later will return very quickly. For example, the sub-trie for prefix `a.b.c/24` can have a bitmap of 2^8 bits, with the bit at index d corresponding to IP address `a.b.c.d`.

If we also need to update the bandwidth consumption information for the IP address (or its prefix), we will need to access its leaf node. To still have a speedy *put* process for duplicate IP addresses, we replace the bitmap with an array of pointers; in other words, instead of setting a bit to `1`, we insert a pointer to the leaf node into the array. In the above example, the pointer at index d will be either *null* or point to the leaf node for IP address `a.b.c.d` (or its prefix).

4.6 PathFinder Proxy

4.6.1 Addressing Design Requirements. The purpose of a PathFinder proxy is to act as an intermediary between PathFinder clients and PathFinder monitors and help the clients learn the footprints of the traffic toward a DDoS victim. Since there can be multiple ASes on the AS-path of the traffic, when the PathFinder monitors of such ASes report the AS-paths of the traffic, the paths they report may overlap. The proxy must determine which AS-path includes the largest number of ASes. Furthermore, if a source-based traffic footprint is requested, these monitors may also store and report the same IP address or prefix. The design of the proxy should resolve the potential conflict between monitors regarding the same IP address or prefix. Finally, the proxy's design should be cost-aware. Since the proxy does not contact monitors unless requested by a client, clearly an on-demand mode is best for the proxy.

4.6.2 Setup. Every PathFinder proxy will make itself available to potential PathFinder clients. If a PathFinder client needs PathFinder service, it will register itself at a proxy, including setting up all necessary security credentials. The client then can send a request to the proxy when it needs to obtain the traffic footprints of a DDoS victim.

We assume the proxy has a list of PathFinder-participating ASes (which the proxy can obtain, for example, through a web page). Further, it knows how to locate the PathFinder monitor of each AS, as described in Sec. 4.4.2.

4.6.3 Operation. Like any PathFinder monitor, every proxy also operates on demand. Once a proxy receives a request from a client, it will verify if the request is authentic and valid or not, and if so, learn who the victim is from the request and forward the request to monitors at PathFinder-participating ASes (or a subset of them if specified in the request). The proxy then receives the PFLogs from monitors via the streaming mechanism (Sec. 4.7), processes them (including merging PFLogs to save storage and transmission overhead), and forwards PFLogs to the client (also via the streaming mechanism).

If the footprints do not need to be source-based, each monitor will function in source-agnostic mode and the proxy will receive PFLogs from each monitor that contain the AS path from the monitor's AS to the victim, as well as bandwidth consumption information if requested. The proxy then merges the PFLogs. Specifically, it adds the path to a path pool, with two exceptions: (1) If the path is just a part—i.e., a **sub-path**—of another path in the pool, the proxy can ignore this path. (2) Conversely, if a path from the pool is a sub-path of this path, the latter will replace the former in the pool. As a result, the proxy will learn a set of AS paths to the victim, and can return them to the client, together with the bandwidth information if requested.

However, if the footprints need to be source-based, the proxy will construct a local PFTrie by merging the PFTries it receives from monitors. For each leaf node of the PFTrie from each monitor, which represents an IP address or prefix `S` that the monitor has captured, the proxy will store `S` in its local PFTrie, following the same *put* process described in Sec. 4.5.1. Furthermore, assuming the monitor's AS is `AS k`, the proxy also marks the leaf node that represents `S` with `k`, to indicate the AS-path of traffic from `S` to the victim is the AS-path from `AS k` to the victim. However, if `S` is already in the local PFTrie, the proxy will retrieve the marked AS number of the leaf node for `S`, say `o`, and mark the leaf node for `S` with either `k` or leave it as `o`, whichever one is upstream of the other. As a result, the proxy then builds a local PFTrie that contains the most complete AS-path from `S` to the victim, and forwards it to the client.

4.7 PathFinder Streaming

A DDoS defense system is more effective if it can respond to a DDoS attack at an early stage, which is only possible if it obtains necessary information about the attack (such as DDoS traffic footprints) with little delay after the attack starts. The methods described in Secs. 4.4 and 4.6 allow a PathFinder monitor and proxy to collect PFLogs (i.e., DDoS traffic footprints). Yet, they do not address *when* a monitor should transmit PFLogs to a proxy or a proxy transmit them to a client to achieve the most timely notification of attack characteristics. PathFinder addresses this issue using a **streaming mechanism**. We first discuss what factors to consider in designing the mechanism, then describe our design of the PathFinder streaming.

4.7.1 Factors to Consider. There are two primary, orthogonal factors to consider when deciding *when* to transmit PFLogs. The first factor is the *preferred interval or frequency* for a PathFinder monitor or proxy to periodically offload PFLogs, as indicated by the DDoS defense system. On one hand, a short interval (i.e., a high frequency) is better, as short intervals allow the DDoS defense system to use information in close to real time. However, short intervals potentially lead to small updates that only contain footprints from a brief time window. Also, shorter intervals incur a higher total network bandwidth overhead. Further, frequent updates may be redundant due to traffic locality and cause each update to more likely overlap with previous updates. On the other hand, a long interval (i.e., a low frequency) would cause a long delay in informing the DDoS defense system about the ongoing DDoS attack. Also, using a long interval may fail to detect certain DDoS attacks; for example, one may need an interval of 200 ms to detect a pulsing attack such as CICADAS [88].

Another factor to consider is the *limited size of the PFLogs buffer* that each PathFinder monitor or proxy allocates to store the PFLogs for its clients. If a client is under source-agnostic mode, its PFLogs (i.e., AS paths and bandwidth information) will be of a small size and a buffer of a fixed small size will be sufficient. On the contrary, if a client is under source-aware mode, the monitor or proxy will be maintaining PFTries in order to log source-based traffic footprints for this client, which can become fairly large. The monitor or proxy then needs to dynamically allocate the buffer space for this client. As it collects more traffic footprints and stores them in the buffer, if the buffer is becoming full it can try to allocate more space to store PFLogs. However, it may not succeed in allocating new space every time, especially if there is an enormous amount of traffic footprints for the client, the interval to offload PFLogs is relatively long, or there are also many other clients to serve simultaneously. In this case, rather than replacing some PFLogs in the buffer to lose these PFLogs, it is preferable to offload PFLogs of this client or other clients to make more space.

4.7.2 Streaming Design. We use a receiver-driven approach to have the client set the preferred interval for receiving streamed PFLogs. When a proxy or a monitor receives a request for collecting traffic footprint for a client, it also learns the preferred interval of transmitting PFLogs for the client, which we call the **streaming interval**. Further, if the client is under source-agnostic mode, it will allocate a fixed-sized buffer for PFLogs. Otherwise, the client is under source-aware mode and the proxy or monitor then dynamically allocates a PFLogs buffer for the client. The proxy or monitor then transmits PFLogs of the client at the streaming interval for the client, which we call **in-sync streaming**. PFLogs transmitted then makes space for new PFLogs in the next streaming interval. If a monitor or proxy cannot allocate more buffer space for a client under source-aware mode, but the streaming interval for the client is not up yet, it also immediately transmits enough PFLogs of the clients with the most PFLogs to free half the buffer space. (These clients may or may not include the client in question.) We call this streaming **out-of-sync streaming**. (PathFinder also supports a third streaming method called on-demand streaming, which is not the focus of this chapter; basically, at any time if needed by DDoS defense, a client may also request its proxy to transmit the current PFLogs to the client, and the proxy in turn can request PFLogs from monitors, thus causing on-demand streaming.)

A PathFinder client under source-aware mode thus may receive PFLogs via in-sync or out-of-sync streaming from its proxy, which in turn may receive PFLogs via multiple in-sync or out-of-sync streamings from monitors. While every streaming above uses an independent channel and can be either in-sync or out-of-sync, it is very likely that if the streaming to the client is out-of-sync, some or many streamings to the client's proxy are also out-of-sync. When out-of-sync streaming happens, although the client can continue to use the current streaming interval, as discussed in Sec. 4.7.1, it is important for the client to use a reasonable streaming interval in order to conduct timely, effective defense of DDoS. However, it is often hard for the client to know how long its streaming interval should be, and the dynamic nature of DDoS traffic only makes it harder.

PathFinder allows its client under source-aware mode to either use a fixed interval for streaming, i.e., **fixed streaming**, or adaptively adjust its streaming interval, i.e., **adaptive streaming**. Via the adaptive streaming, a client can choose an initial streaming interval, such as a default value of 1 second, and then adjust this interval using the following two complementary strategies.

The first strategy uses the number of new DDoS sources from an in-sync streaming to adjust the interval. Each time an in-sync streaming occurs after a current interval, the client inspects PFLogs from the interval to calculate the number of new DDoS sources per time unit, i.e., the current rate of new DDoS sources, or S_{curr}. It then uses an exponential moving average method to estimate the average rate of new DDoS sources, or S, using the follow equations:

$$S = S_{curr} \tag{4.1}$$

$$S = \alpha S_{curr} + (1 - \alpha)S \tag{4.2}$$

where Equation (4.1) is the estimate after the first in-sync streaming, and Equation (4.2) is the estimate after an in-sync streaming, with α being configured to a suitable value, such as 0.5, by the client. If the client wishes to receive a specific number of new DDoS sources in the next interval, say N, it then can set the next streaming interval as $\frac{N}{S}$.

The second strategy uses the occurrence of out-of-sync streaming as a signal to indicate that the buffer space at the proxy and probably some monitors are full and that the client should use a shorter streaming interval. So, whenever an out-of-sync streaming happens, a client will reset its streaming interval to be half of the previous streaming interval value. Furthermore, the client will also record this interval value as a threshold, such that if later the first strategy is applied to adjust the streaming interval, the new interval value should not surpass this threshold.

4.8 On-Demand Zooming

Sometimes a DDoS attack can be of a very large scale where the DDoS traffic may come from millions of IP addresses. If a PathFinder client wishes to capture DDoS traffic footprints of such an attack under source-aware mode, PathFinder monitors and proxies may have to storage and transmit a high volume of PFLogs, while the PathFinder client may also be overwhelmed by the volume of PFLogs it receives. The latency of streaming will also become higher, potentially lengthening the DDoS defense response time. The PFTrie optimization techniques we introduced in Sec. 4.5.2 may help, but not always. For example, as there is no guarantee in DDoS traffic locality, leaf nodes of PFTrie may not be aggregatable; or, if the attack involves a lot of unique source IP addresses, many new IPs need to be recorded in the PFTrie, thus not causing duplicate traversals that can be avoided.

PathFinder addresses this issue using an **on-demand zooming mechanism** that can significantly reduce the size of PFLogs. Specifically, when a PathFinder client initially issues a request for source-aware PFLogs, it can specify a coarse granularity for recording the traffic footprints. Only if the client needs to know the DDoS sources at a finer granularity later will the client *zoom in* and request PFLogs of DDoS sources at a finer granularity. It can repeat this on-demand zooming process until it is satisfied with the granularity. Similarly, a client can also zoom out and begin to request PFLogs at a coarser granularity.

For example, initially, a client can ask for traffic footprints of /16 IP prefixes. As it receives PFLogs of /16 IP prefixes, because a /16 IP prefix consumes the most network bandwidth, it decides to zoom in on this prefix to collect PFLogs about it at a finer granularity. It thus issues a new request of PFLogs; in addition to still requesting PFLogs of other /16 IP prefixes, it also requests PFLogs of /20 sub-prefixes of the /16 IP prefix in question. The on-demand zooming process can repeat recursively; for instance, the client may further zoom in on /24 sub-prefixes on some of the /20 sub-prefixes, or subsequently on IP addresses of certain /24 sub-prefixes.

The benefits of on-demand zooming are obvious. By only zooming in on IP prefixes of interest, monitors and proxies could store PFTries at a coarse granularity for most IP Prefixes. The depth of PFTries could be shortened, and the size of PFTries could become smaller. For instance, before zooming in on any IP prefixes, a PFTrie storing $/p$ IP prefixes will be no more than p levels; compared to a 32-level PFTrie storing IPv4 addresses (i.e., /32 IP prefixes), it has a significant storage and transmission overhead reduction of $O(2^{16})$. After zooming in on certain IP prefixes, the savings on storage and transmission overhead will become less. However, unless the on-demand zooming mechanism is repeatedly used and eventually results in a PFTrie that records every IP address, which rarely happens, the savings will continue to be significant.

On-demand zooming also increases the chance to apply the PFTrie optimization methods from Sec. 4.5.2. Assume a PFTrie leaf node representing an `X0`$/p$ IP prefix, where `X` represents the first *p-1* bits of the prefix. First, it is more likely to optimize a PFTrie via the bottom-up aggregation of leaf nodes (Sec. 4.5.2.1). As long as there is another packet from IP prefix `X1`$/p$, we can aggregate this leaf node with a new leaf node representing `X1`$/p$ and replace them with their parent node `X`/*p-1* as a new leaf node. Clearly, the smaller p is, the bigger address space we have with `X1`$/p$, thus the more likely for this optimization to happen. Second, the smaller p is, the bigger address space the leaf node represents, the more likely to have more packets from this space to avoid duplicate traversal of the PFTrie, another optimization method (Sec. 4.5.2.3).

4.9 Evaluation

4.9.1 Goals.

We now show our evaluation of PathFinder in terms of the following:

1. *Benefits of PathFinder for DDoS defense:* Since PathFinder footprints include path and traffic information, any DDoS defense system that deploys filters inside the network to discard DDoS traffic can take advantage of the footprints to make better filter deployment decisions. We built a DDoS attack

and defense simulation to study how PathFinder can benefit a DDoS defense system and make it more effective. We look at four different strategies of placing DDoS traffic filters and show that a DDoS defense system—when utilizing PathFinder—uses much fewer resource and achieves a much higher level of success. We then show PathFinder is beneficial even when only partially deployed.

2. *Speed and overhead of PFTrie:* At the core of PathFinder are the PFTrie-based operations at each monitor and proxy, particularly the *put* process. Therefore, we evaluate the time to store an IP address or prefix `S` in a PFTrie under two scenarios. In one scenario, `S` is new and the PFTrie needs to be updated with a set of new nodes, including the leaf node that represents `S`. In another scenario, `S` is in the PFTrie, so the *put* process will check and return the current leaf node that represents `S`. We call these two scenarios **put_a_new** and **put_an_old**, respectively. Further, we evaluate the memory overhead of the PFTrie at a monitor or proxy when producing source-aware footprints and the network overhead when transmitting a PFTrie.
3. *Benefits of PathFinder streaming:* PathFinder uses a streaming mechanism to enable its clients, i.e., DDoS defense systems, to obtain the DDoS traffic footprints quickly. Via our simulation, we evaluate the delays and network overhead when using streaming to obtain traffic footprints. In particular, we show how a client under source-aware mode can use adaptive streaming to achieve both timely traffic footprint delivery and low network overhead.
4. *Benefits of PathFinder zooming:* In addition to designing optimization methods to save the overhead of PFTries, we further designed the zooming mechanism that can keep PFTries at a potentially much smaller size and only

grow PFTries on demand. Using synthetic traces with up to 16 million IP addresses, we show PFTries can indeed not only dramatically reduce their size if zooming is employed, but also keep their size small if some prefixes are zoomed in.

5. *Traffic capturing strategy:* Besides relying on the streaming and zooming mechanisms to save the cost in capturing the footprints of DDoS traffic for a client under source-aware mode, the monitor at a PathFinder-participating AS may consider capturing only a sample of the traffic *if* doing so would still capture most, if not all, the source addresses or prefixes of the traffic. Furthermore, the monitor may lack modern hardware or software support (e.g.wiretapping is infeasible) or be constrained by the policy of the AS (e.g.only sampled mirroring is allowed), so it can only provide sampled traffic, anyway. We therefore investigate how long a monitor should observe the traffic and whether it can collect a sample of the traffic rather than all the traffic, while still learning most IP sources of the traffic.

4.9.2 Benefits of PathFinder for DDoS Defense. To quantify the benefits of PathFinder for DDoS defense, we first define the model of DDoS attack, then compare the results of a DDoS defense system with and without PathFinder, and finally study the partial deployment of PathFinder. Rather than using actual measurements over the Internet, this study is simulation based because our evaluations are at a large scale with more than 60,000 ASes, where the victim may be in any stub AS, DDoS bots may be distributed throughout all stub ASes, and PathFinder may be deployed at certain ASes, all of which are extremely hard to arrange on the real Internet. Simulation allows us to experiment with different configurations (e.g.different victim ASes) and collect results of many different scenarios (e.g.with or without PathFinder, with or without IP spoofing, different PathFinder deployment rates, etc.).

4.9.2.1 DDoS Attack Model. The DDoS attack model includes the following components:

1. **DDoS victim (target AS).** Every DDoS victim is located as a random tier-3 AS. We also call this AS the *target AS*. The victim can at most handle 10 Gbit/s incoming traffic. We use tier-3 ASes because DDoS attacks often target the services or applications hosted in such networks (e.g., educational networks, small data centers). Their limited link bandwidth makes them subject to most DDoS attacks reported [58]. On the other hand, tier-1 and tier-2 ASes, whose business is primarily to forward traffic and link capacity is generally very high [155], can hardly be threatened by DDoS attacks [31, 96].
2. **DDoS bots.** The DDoS attack is based on high-volume traffic from DDoS bots where each bot has a fixed uplink bandwidth of 25 Mbit/s. Since a batch of bots will no longer be effective once filters are deployed to filter their traffic, an intelligent attacker would periodically switch to a new batch of bots to launch its attack. We define the *attack cycle* as the time window for every batch of bots used by the attacker. There are a total of 100,000 bots residing in stub ASes. (An AS may be either a stub AS, which is a source or sink of traffic and announces IP prefixes that it is in charge of, or a transit AS, which forwards traffic for stub ASes.) Except for the target AS, we assign every stub AS a number of DDoS bots that is proportional to the number of IP addresses announced by the AS.
3. **Topology.** For every target AS, we derive an AS-level topology based on the relationships of ASes on the Internet according to CAIDA [25]. We use

AS-level topologies rather than those at the router or PoP level because PathFinder is designed to operate at AS level (Sec. 4.3.1). The AS path from the target AS to every other AS follows the valley-free model [114]. The AS path from an AS to the target AS is either the reverse of the path from the target AS to the AS, or an asymmetric path with 30% probability.

4.9.2.2 DDoS Defense Enhanced with PathFinder. We simulate a DDoS defense as follows. During a DDoS attack, the victim employs a DDoS defense system that attempts to place a DDoS traffic filter at an AS on every path of DDoS traffic. Note that in the real world, the defense system must first detect attack traffic to generate filters, which is a challenging task that requires a separate research study. For the sake of simplicity, we assume the detection accuracy is the same for all defense scenarios in this simulation. To not have the DDoS traffic, as they converge, overwhelm links close to the target AS, the defense only places filters at tier 2 and 3 ASes that are on the first half of the AS paths of DDoS traffic. With PathFinder's help, the defense knows accurately the AS paths of DDoS traffic and the bandwidth consumption information of DDoS traffic along these AS paths. Here, we evaluate two AS selection methods for filter placement: 1) randomly select an upstream AS; 2) select an AS that belongs to top k ASes that carry most of the DDoS traffic.

Otherwise, without PathFinder, the defense has to guess the paths of DDoS traffic. It assumes AS paths are symmetric and regards the AS path from any DDoS bot to the target AS is always the reverse of the AS path from the target AS to the bot. From our model of the topology in Sec. 4.9.2.1, this assumption leads to a 30% miss in placing filters due to path asymmetry. Worse, if a DDoS bot spoofs its source address, the defense will even use a wrong AS path from the target AS to the bot to begin with, leading to a higher rate of misplaced filters; we assume in this case 50% filters are misplaced.

We evaluate PathFinder's benefits under the worst-case DDoS attack scenario. Specifically, the attacker employs an attack cycle that is no greater than the defense response time, which is the time for PathFinder to collect footprints of newly seen DDoS traffic and also for the DDoS defense system to place the filters against the traffic. Thus, before the DDoS defense places filters for the current bots attacking the victim, the attacker already switched to a new attack cycle.

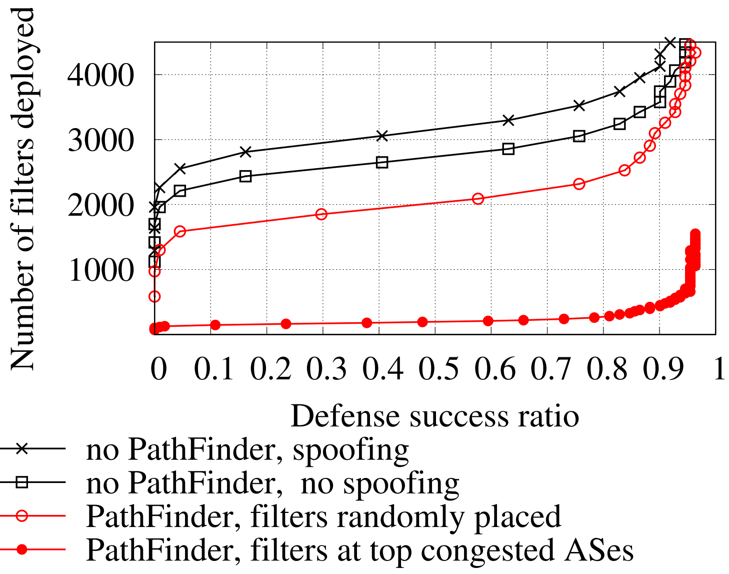


Figure 18. DDoS defense with and without PathFinder.

Figure 18 compares the DDoS defense with or without the help of PathFinder. It shows the number of DDoS traffic filters needed for filtering out different DDoS traffic ratios, i.e., defense success ratio, under four different scenarios. To avoid potential selection bias of target ASes, we conducted 80 different experiments, each with a different target AS, and then averaged the results from all experiments. The DDoS defense system without PathFinder performs much worse than the two cases that use PathFinder. Without

PathFinder, it takes at least 3,500 filters to subdue 90% of the attack traffic and even more (about 4,130 filters) when some DDoS bots employ IP spoofing. However, with PathFinder in place, if applying filters at top congested ASes (which requires AS path and bandwidth information from PathFinder), the victim can survive 90% of the attack traffic with only roughly 500 filters; in the case when the DDoS defense system uses only AS path information from PathFinder and places filters randomly at ASes, the system still uses a much smaller number of filters compared to the two defense cases without PathFinder.

4.9.2.3 Partial Deployment of PathFinder. We studied the partial deployment of PathFinder when not every AS runs PathFinder. If on the path from a DDoS bot to the victim there is at least one AS running PathFinder, we can say this bot is "covered" since the DDoS traffic footprints from this bot will be reported by PathFinder; otherwise, the bot is not "covered". Therefore, to evaluate the coverage provided by PathFinder when it is only partially deployed, we can measure the percentage of DDoS bots that are "covered", i.e., traffic footprints coverage by PathFinder.

Not every AS on the Internet is equal. It is difficult for tier-1 ASes, which usually could span an entire country or even beyond, to deploy PathFinder throughout its entire network due to its sheer size and high cost. Tier-3 ASes are small and much easier to deploy and run PathFinder, but given the most ASes on the Internet are tier-3 ASes at the edge of the Internet, and they can only report footprints of traffic originated from their network, PathFinder has to be deployed at a substantial amount of tier-3 ASes to have a clear impact. The best trade-off is to look at deploying PathFinder only at tier-2 ASes, which are not only easier to deploy and run PathFinder but can also report footprints of DDoS bots located at a multitude of ASes.

Therefore, we measured the traffic footprints coverage by PathFinder using the DDoS attack model described in Sec. 4.9.2.1. We experimented $5,000$ different topologies, each with a different target AS and a range of PathFinder deployment rates among tier-2 ASes. Figure 19 shows that when 55% or more tier-2 ASes run

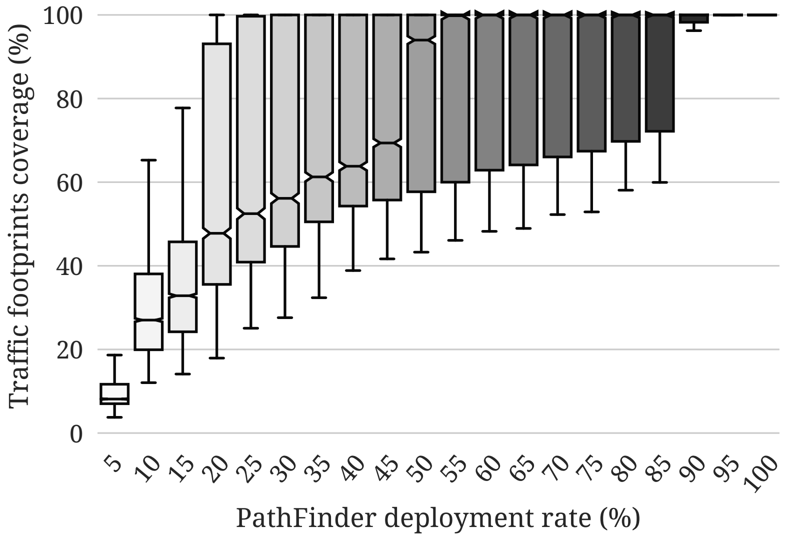


Figure 19. Partial deployment of PathFinder at tier-2 ASes.

PathFinder, virtually 100% of the DDoS bots (i.e., their traffic footprints) will be covered for half of the experimented topologies; i.e., there is at least one AS that captures all traffic footprints of each traffic destination AS. Note that given about 25 tier-1 ASes, 10,000 tier-2 ASes, and 58,000 tier-3 ASes on the Internet, 55% tier-2 ASes is about 8.1% all ASes [25]. Even when it is less than 55%, PathFinder can be very effective. For example, even when only 20% of tier-2 ASes (i.e., 2.9% of all ASes) run PathFinder, half of the evaluated topologies will have at least 48% DDoS bots covered, and a quarter of evaluated topologies will have at least 94% DDoS bots covered.

4.9.3 Speed and Overhead of PFTrie.

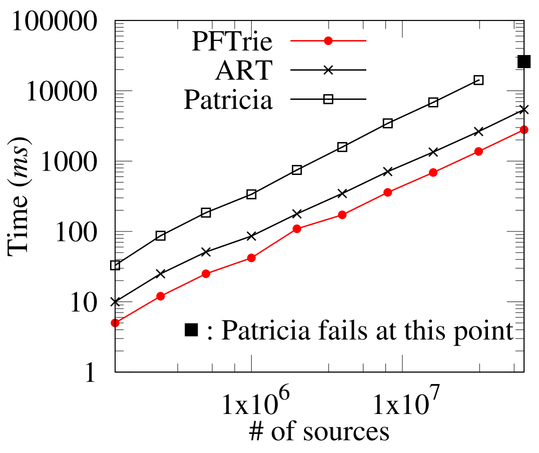
4.9.3.1 Experiment Setup.

To evaluate the PFTrie speed and its memory overhead, we used a desktop with Intel i7-4790 at 3.6 GHz with an 8-MB L3 cache and a 32-GB RAM at 1600 MHz. We implemented the PFTrie in *C*, and used the *Clang* compiler with the optimization level 2 to compile the code. We also created 50 synthetic traffic traces that contain 125 thousand to 64 million IP addresses; for each size we created five traces with different levels of source address locality, with 0%, 25%, 50%, 75%, and 100% addresses, respectively, that belong to the same IP prefix and can be aggregated. We choose not to use publicly available DDoS traces as their attack volume/sources are too small of a scale to benchmark the scalability of PFTrie.

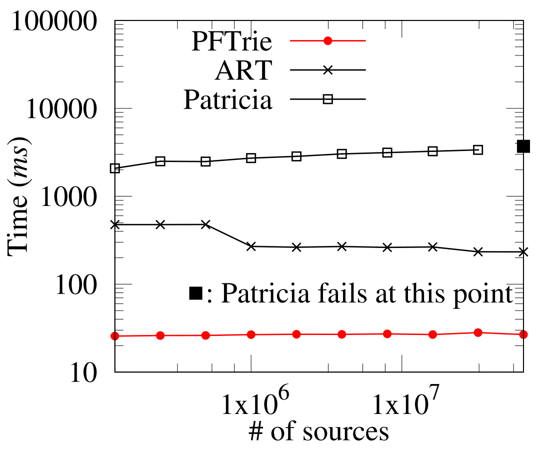
4.9.3.2 Speed of PFTrie.

We compare PFTrie's performance against Adaptive Radix Tree (ART) [104] and the well-known Generalized Prefix Tree [133], also called Patricia Trie, under the two scenarios, **put_a_new** and **put_an_old**, as defined in Sec. 4.9.1. We use all the 50 synthetic traces that contain different orders of magnitude numbers of IP addresses to evaluate the three different data structures under stress.

Fig. 20a shows the comparison results under the put_a_new scenario. It reports the time for ART, Patricia Trie, and PFTrie to store all the IP addresses in a trace with 25% address locality as new addresses. Our results with traces of different locality are similar. For every synthetic trace size ranging from 125,000 to 64 million source addresses, when storing a new IP address or prefix, PFTrie always outperforms ART and Patricia. For example, to store 16 million IP addresses, it takes more than 1300*ms* for ART and about 6,600*ms* for Patricia but it only takes around 700*ms* for PFTrie. In general, PFTrie spends 50% less time than ART to



(a) put_a_new scenario.



(b) put_an_old scenario.

Figure 20. PFTrie speed.

store the same number of IP addresses. Even to store 64 million IP addresses, it takes only 2.93s. Note that Patricia fails when presented with this many addresses.

Fig. 20b shows results under the put_an_old scenario for performing 15 million *put* processes that stores an IP address *already* stored. These IP addresses are from the same trace as the one used in Fig. 20a. Here, as there are no new nodes to be inserted into a PFTrie, the time needed by a PFTrie is virtually constant at about 27.0ms, much less than that in the put_a_new scenario; e.g.we can deduce with 64 million IP addresses, it would be about 115ms as opposed to 2.93s in the put_a_new scenario. Moreover, the time is also less than that of ART and Patricia, and PFTrie is at least 10 times faster than ART in every case. This speed is because of the PFTrie optimizations we introduced, including the top-down collapse of prefixes (Sec. 4.5.2.2) and the avoidance of duplicate traversals (Sec. 4.5.2.3).

While the PFTrie operations are a combination of the two scenarios, from various real-world traces we notice that the put_an_old scenario is more frequent

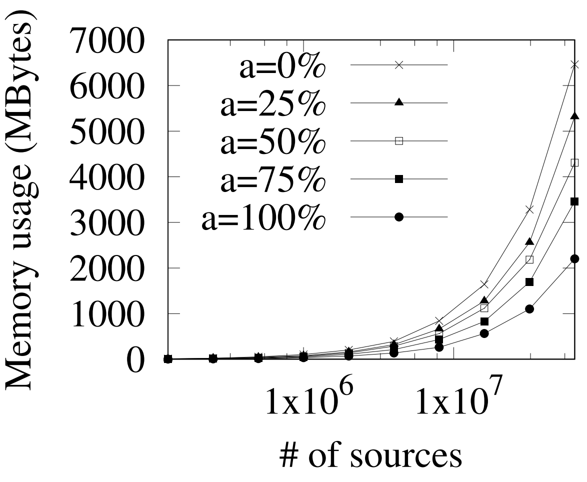


Figure 21. PFTrie memory overhead across five different source profiles. Each profile has a different percentage of aggregatable addresses (a).

than the put_a_new scenario. For example, in the Booter 3 DDoS trace [183, 168], the put_an_old scenario will happen 369 times more than the put_a_new scenario.

4.9.3.3 Overhead of PFTrie. We are particularly interested in the memory cost of PFTrie when there are millions of IP source addresses. Using synthetic traces that include a huge number of IP source addresses, we evaluated the memory usage for each number of sources under five different profiles, as shown in Fig. 21. We can see the logarithm of the memory cost is basically a linear function of the logarithm of the number of sources, and overall the memory cost is manageable under all five profiles. Moreover, a profile with a higher address locality can have much lower memory cost, due to the optimization via bottom-up aggregation of PFTrie leaf nodes (Sec. 4.5.2.1). Because of the tree nature of PFTrie, the memory cost complexity for storing 2^n addresses in a PFTrie is $O(2^n)$ with n levels of nodes, but if it shrinks to k levels, the memory cost will become $O(2^k)$, a reduction of $O(2^{n-k})$ times.

We also evaluated the network overhead with 25 different AS-level Internet topologies. We generated these AS-level topologies using the same method described in Sec. 4.9.2.1, with each topology using a different random tier-3 AS as a target AS. For each topology, we further populate it with one million IP addresses across all the ASes, where the number of IP addresses assigned to each AS is proportional to its IP address space size.

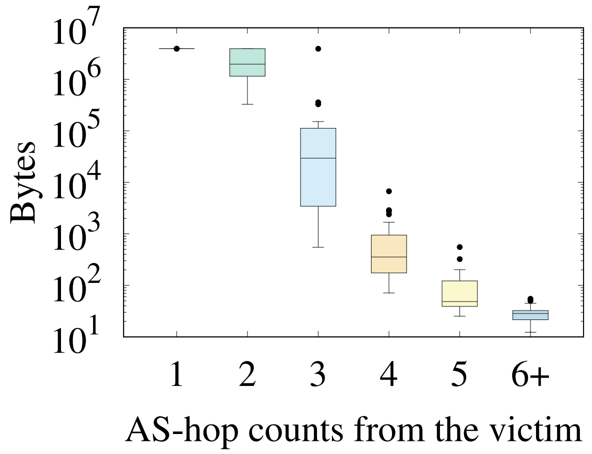
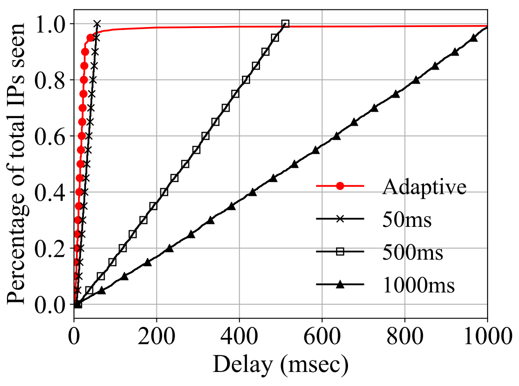


Figure 22. The boxplot of network overhead in transmitting PFTries of 1 million source addresses.

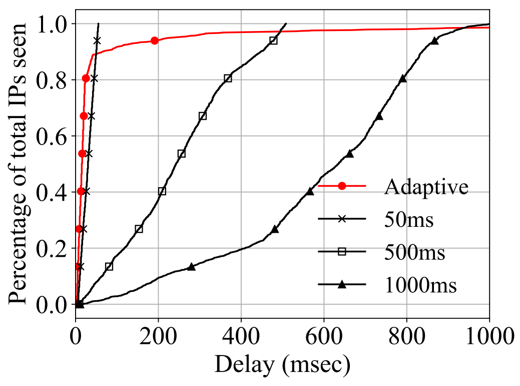
Fig. 22 shows the network transmission overhead for an AS to transmit its PFTrie to a proxy. Clearly, the further away an AS is from the victim, the smaller the network overhead it introduces. The AS that is the last hop to reach the victim would see traffic from all one million IP addresses, thus incurring the largest overhead, which is only about 3.9 MB.

4.9.4 Benefits of PathFinder Streaming. We ran simulation studies over an inferred AS-level topology to measure the delay and overhead with PathFinder streaming, including how effective and efficient the adaptive streaming design is. Below we describe our simulation model and results.

4.9.4.1 Streaming Simulation Model. The streaming simulation model includes all PathFinder components (monitor, proxy, client) and an inferred AS-level topology generated using the method described in Sec. 4.9.2.1. While the overall Internet connection speed has become faster over the years, we use conservative numbers in our simulation. Specifically, for each link on the topology we assume it has a random delay between $1ms$ and $10ms$ and a fixed bandwidth of 10 Mbps. (According to the 2019 report from Opensignal [149], the connection speed over a *mobile network* is 10+ Mbps in 67 out of 87 surveyed countries and 5+ Mbps in more than half of the remaining 20 countries.) Moreover, we use two Booter DDoS traces ([183][168]) to drive the streaming simulations. We map the IP addresses in these traces to ASes (or PathFinder monitors) in our simulation.



(a) DDoS trace: Booter 5



(b) DDoS trace: Booter 9

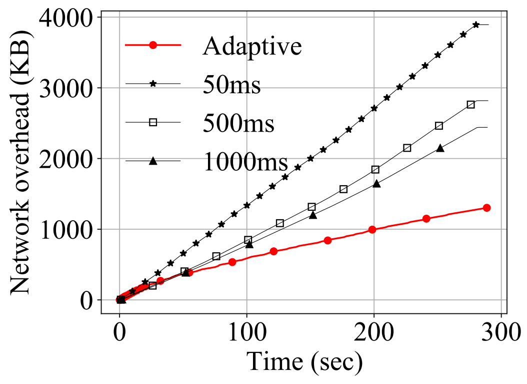
Figure 23. Delay CDF with fixed and adaptive streaming.

4.9.4.2 ***Delay with Streaming.***

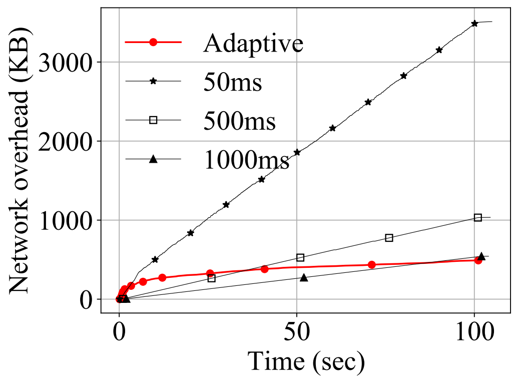
We now evaluate the delay with both fixed streaming and adaptive streaming, i.e., the time it takes for a client to learn the IP address of a DDoS bot after the first DDoS packet departs from the bot toward a victim.

Figures 23a and 23b show the delay distribution over Booter 5 and Booter 9 DDoS traces, respectively. In both figures, adaptive streaming outperforms fixed streaming at three different fixed intervals. For both attack traces, adaptive streaming delivers more than 90% of the DDoS IP addresses in less than 50 milliseconds. The adaptive streaming tends to have longer delays for the later half of the DDoS traces as the traces contain many fewer new sources in their second half (the attacker has a limited number of bots for each attack); this behavior is consistent with the style of adaptive streaming as it is not aggressive in learning a trickle of new DDoS IP addresses.

4.9.4.3 Network Bandwidth Overhead with Streaming. We further evaluate the network bandwidth overhead over an extended period of time with both fixed streaming and adaptive streaming. Figures 24a and 24b show the cumulative network bandwidth overhead from adaptive or fixed streaming at each time point using Booter 5 and Booter 9 traces, respectively. Clearly, although the fixed streaming with a short interval ($50ms$) has a small delay (Figures 23a and 23b), it incurs a network bandwidth overhead that is approximately 4 and 7 times, respectively, more than the adaptive streaming. The fixed streaming with a medium and long intervals ($500ms$ and $1000ms$) incur less bandwidth overhead than the fixed streaming with a short interval ($50ms$), but such a bandwidth overhead is at the cost of the delay in obtaining traffic footprints as shown above and is still more than that of the adaptive streaming.



(a) DDoS trace: Booter 5



(b) DDoS trace: Booter 9

Figure 24. Network cumulative bandwidth overhead with fixed and adaptive streaming.

4.9.5 Benefits of PathFinder Zooming.

We now show how much memory overhead that the zooming mechanism can help reduce. (We have shown the memory cost of PFTries when zooming is not in effect in Sec. 4.9.3.3.) We use the synthetic traces whose source address locality is 0% (worst case).

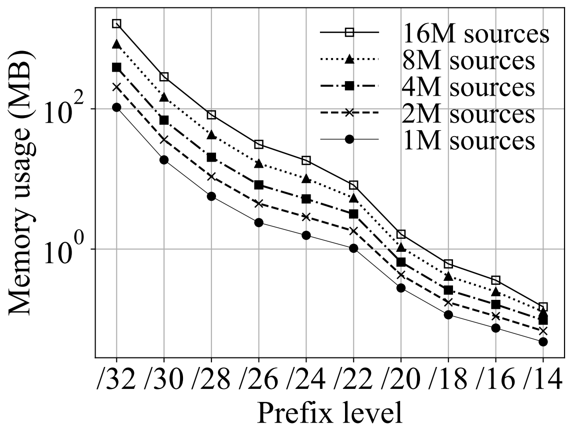


Figure 25. Memory overhead with zooming (without zooming in on any prefix).

Figure 25 shows the memory overhead for storing PFTries when the zooming technique is used with varying prefix levels and different numbers of sources but *without* zooming in on any prefix. In all cases, we can see the memory overhead decreases exponentially as we reduce the monitored prefix level. For example, it costs 1.6 GB of memory space to save 16 million IPv4 addresses (/32) while it costs only 0.36 MB of memory space to save the same traces at /16 prefix level. We also notice that as the prefix level decreases the slope of each line also varies; this is due to the different degree of bottom-up aggregation optimization of PFTries at each prefix level and a bigger slope means a higher rate of aggregation and vice versa.

We further measured the memory overhead when a PathFinder client requests to zoom in on certain prefixes that contribute more traffic with more DDoS bots than other prefixes. Such disproportional distribution can be seen in, for example, the Mirai DDoS attack [8]. We used a synthetic trace with totally 16 million IP addresses and a 0% source locality (i.e., a=0%). With totally 2^{16} (i.e., 65536) /16 IP prefixes, in this trace the number of IP addresses allocated to each /16 prefix follows a power law distribution, with a small percentage of /16 prefixes crowded with a large portion of IP addresses; for example, top 100, 200, 400, 800, and 1600 crowded prefixes respectively cover 19%, 33%, 56%, 80%, and 96% of all 16 million IP addresses.

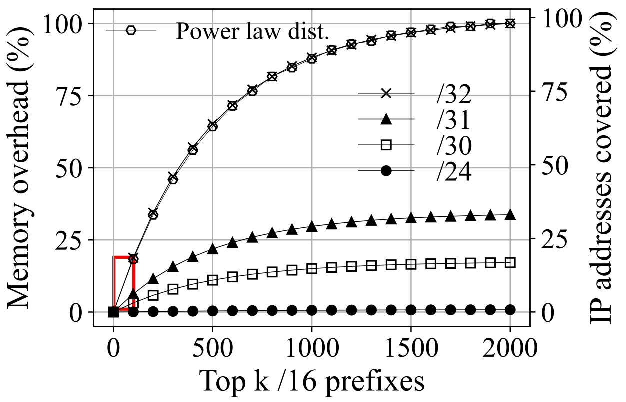


Figure 26. Memory overhead as a percentage of the baseline and IP addresses covered, both with top k /16 prefixes zoomed in.

Figure 26 shows the results. Define the baseline as the memory usage of the PFTrie that stores every IP address from this trace (i.e., zooming is *not* in place), which is 1.6 GB as seen in Figure 25. Figure 26 shows the memory overhead as a percentage of the baseline when the client, which initially monitors /16 prefixes, decides to zoom in on top-k crowded /16 prefixes to all their /24, /30, /31, and /32 sub-prefixes. We can see that when the client zooms in on a small number of prefixes to any prefix level, the memory overhead is quite small; for example, if zooming in on top 100 prefixes, which is only 0.15% of all prefixes but with 19% of the 16 million IP addresses, the memory overhead is 18.8% in the worse case when reaching all /32 prefixes (as shown in the red box in Figure 26). Also, when the client zooms in on *all* prefixes to /31, /30, or /24 prefix levels, the memory overhead will also be fairly small at 33.8%, 17.1%, and 0.79%, respectively. Overall, the flexibility offered by zooming allows the client to choose which prefixes and to which prefix levels to zoom in; except for rare cases (such as zooming in on all prefixes to every single IP address level), the memory overhead saving with zooming is significant, often at multiple orders of magnitude.

4.9.6 Traffic Capturing Strategy Analysis. Rather than monitoring a DDoS attack over its entire life cycle and capturing every single packet toward the DDoS victim, a monitor may only observe the DDoS traffic for a time window and also only sample the traffic. However, doing so could cause certain source addresses in the traffic not to be captured, reducing the traffic capturing accuracy to below 100%. The question is, is it possible to maintain a high accuracy with a certain size of monitoring window and a certain traffic sampling rate?

Dataset	# of attack sources	DDoS bandwidth (Mbps)	new/duplicate sources
Booter 1	4486	700	1/72
Booter 2	78	250	1/230
Booter 3	54	330	1/370
Booter 4	2970	1,190	1/14
Booter 5	8281	6	1/2.5
Booter 6	7379	150	1/5.5
Booter 7	6075	320	1/3.6
Booter 8	281	990	1/157
Booter 9	3779	5,480	1/57

Table 5. The Booter DDoS traffic traces (9 datasets) [183, 168]. (The last column shows the ratio of the new source addresses versus the duplicate addresses in the trace.)

We measured the accuracy of capturing the source addresses from a real DDoS traffic trace. Specifically, we use the Booter 9 traces for this experiment; it has the largest attack bandwidth among all Booter attack datasets shown in Table 5. By varying the size of the monitoring window from 0.2 to 3.0 seconds and the traffic sampling rate from 0.5% to 20%, we obtained the results depicted in Fig. 27. We can see that while within 3 seconds the sampling rates up to 2.5% can

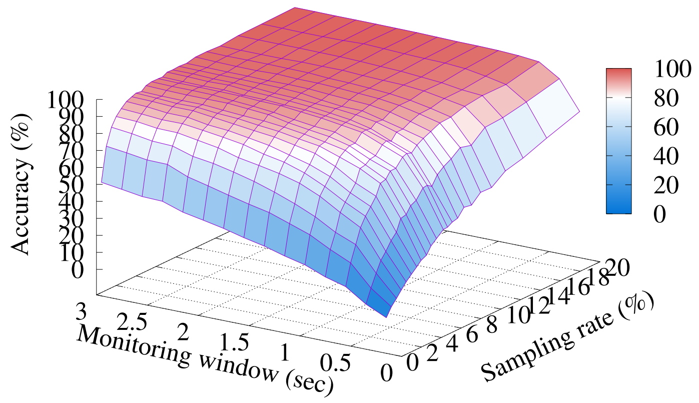


Figure 27. Accuracy of capturing the source addresses (Booter 9 DDoS traces).

never reach an accuracy of more than 90%, starting from a sampling rate of 3%, the accuracy is already as good as 90% when monitoring the traffic for 3 seconds. On the other hand, if the monitoring window needs to be as short as possible, the sampling rate has to be higher; e.g., if only monitoring traffic for 0.4 seconds, the accuracy can be as high as 85% with 18% sampling rate.

As the sampling rate and the window size can both affect the accuracy, we further compare the accuracy with a faster sampling rate in a shorter window and the accuracy with a lower sampling rate but a longer window. Fig. 28 shows that when the product of the window size and the sampling rate is the same, a higher sampling rate in a shorter window consistently performs slightly better than its counterpart.

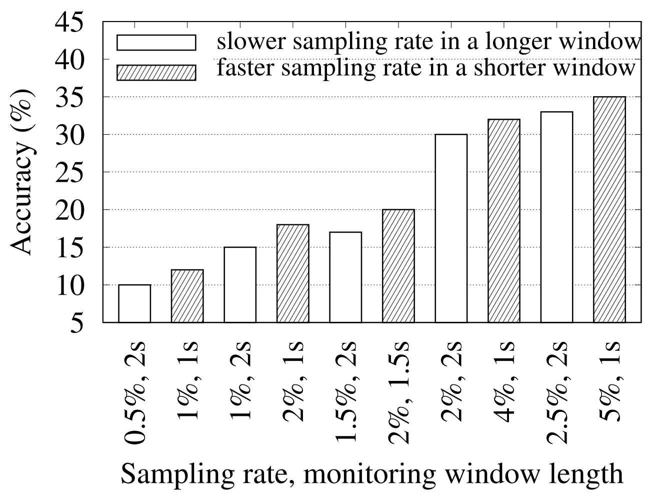


Figure 28. Accuracy of capturing sources with a faster sampling rate in a shorter window (Booter 9 DDoS traces).

4.10 Discussions

PathFinder is an approach to obtaining the DDoS traffic footprints at the Internet scale, and we have made many design choices to provide a line-rate, cost-effective, and deployable solution. Nonetheless, some issues remain to be discussed.

4.10.1 System Scalability. Since the PathFinder system is stateful, it must consider scalability issues related to, for example, the memory and input/output resources. For example, an adversary can employ IP spoofing, generate traffic from arbitrary IP addresses, and overwhelm the system if it operates in source-aware mode. In fact, PathFinder has embraced a suite of different mechanisms to make itself scalable. We have designed three different methods to optimize a PFTrie (Sec. 4.5.2). Also, a PathFinder client can always utilize the zooming mechanism to consume least possible memory and network resources and only zoom in on prefixes of interest, such as those with a large number of IP addresses or voluminous traffic. Moreover, a monitor or a proxy can always use out-of-sync streaming to proactively free more buffer space for more footprints later. Finally, in case out-of-sync streaming cannot work either due to network congestion, the monitor (or the proxy) could prune its PFTrie bottom up level by level as needed; for instance, it could prune all leaves at level 32 to shrink the PFTrie to have 31 levels, or even less by repeating the pruning of leaves at the current level, where the traffic footprint information carried by every leaf node is transferred to its parent. The pruning procedure is similar to the bottom-up aggregation of leaf nodes, except that here a leaf is removed even if it is the only child of its parent. A PFTrie with x levels after pruning also looks the same as a PFTrie with zooming at $/x$ IP prefixes in place, except that the former is invoked by a monitor (or a proxy) and the latter is requested by a client. As a result, PathFinder will be able to collect and deliver traffic footprints from an arbitrary number of IP addresses, including when an attacker overwhelms PathFinder with a vast amount of spoofed traffic as mentioned in the example above.

4.10.2 System Security. We assume that the PathFinder system employs state-of-the-art defense mechanisms to protect itself against any security attacks. Using standard security mechanisms such as mutual TLS that enable two parties to authenticate each other and establish a secure channel [26], every PathFinder client can establish a secure channel with its PathFinder proxy, and similarly, every proxy can establish a secure channel with every PathFinder monitor. These channels can then protect the integrity and confidentiality of both requests for DDoS traffic footprints (which are sent from a client to its proxy and then to PathFinder monitors) and DDoS traffic footprints themselves (which are delivered from monitors to proxies and then to clients). Note that PathFinder employs a trust model in which both proxies and monitors are trustworthy. As a PathFinder proxy is to provide service to multiple clients, it is closely monitored and well-protected. And it is safe to assume that PathFinder monitors are not ill-intentioned to provide false fingerprints; otherwise, the AS hosting a malicious monitor is likely to be malicious or compromised as a whole, in which case the attacker is not going to waste his time generating false PathFinder fingerprints as they are not nearly the worst trouble the attacker can cause.

4.10.3 System Deployment. PathFinder addresses a critical missing gap of many DDoS defense systems that often lack sufficient knowledge of DDoS traffic footprints, including attack paths, bandwidth consumption along each path, and the source addresses or prefixes of DDoS traffic. Indeed, while the network community is moving towards a fine-grained DDoS defense style (e.g.BGP FlowSpec [123]), the need of fine-grained DDoS traffic footprint information becomes more urgent. Moreover, PathFinder is deployment friendly. It employs an architecture that is easy to implement and deploy on today's Internet; PathFinder monitors and proxies, in particular, can be easily set up by their AS. The streaming and zooming mechanisms of PathFinder further make it affordable to deploy PathFinder. Also, the PFTrie data structure includes multiple design features to allow a monitor to log traffic at a line rate. Finally, we emphasize PathFinder is beneficial to DDoS defense even when its deployment rate is relatively low (Sec. 4.9.2.3).

4.11 Conclusions

While DDoS attacks have become more frequent and can cause severe damage to services on the Internet, defense against such attacks has often suffered from the lack of relevant knowledge of the DDoS traffic. However, it is challenging to grasp the topological nature of the DDoS traffic while the attack is occurring: the DDoS traffic often originates from many different locations, follows various paths to reach the victim, sometimes carries spoofed source addresses, and can be extremely dynamic. Currently, the best options are various IP traceback or path inference approaches, but they impose stringent demands to run and deploy. (e.g., many IP traceback solutions require inter-AS collaboration to work, and path inference approaches fails to capture the AS-paths that carry spoofed traffic.) We fill this gap by proposing the PathFinder system as a service that a DDoS defense system can use to obtain the footprints of the DDoS traffic to a victim, including specifying many details of the footprints such as whether the source address and/or bandwidth information is needed. In particular, PathFinder embraces an architecture that not only eases its deployment in today's Internet, but also ensures it has a low cost (e.g., its on-demand model) and is fast to meet the line rate of the packets it must capture. It incorporates a PFTrie data structure that introduces multiple design features to log traffic at a line rate, as well as streaming and zooming mechanisms that facilitate the storage and transmission of DDoS footprints more efficiently.

In addition to building a pilot version of the PathFinder system, we have performed substantial performance experiments to characterize its effectiveness, speed, and costs. These experiments show that PathFinder can deliver accurate information about DDoS flows to a site under attack at reasonable costs both to the systems gathering the data and the defense system that consumes it. Our evaluation shows that the PathFinder system is viable: Even with a low-end desktop machine it only takes about 2.93 seconds to store 64 million unique IP addresses, or only 115 milliseconds in total to check if they are previously stored, while the memory and network overhead is manageable and scalable, especially with the help of the streaming and zooming mechanisms.

CHAPTER V

MOVING TARGET ATTACK DESIGN AND ANALYSIS – TOWARD UNDERSTANDING THE INADEQUACIES OF DDOS DEFENSE DESIGN IN REAL WORLD

Chapter IV shows why an ideal DDoS defense system needs rich input to function effectively. This chapter exposes design issues in DDoS defense solutions on the Internet today via adversarial thinking. Specifically, this chapter first presents a DDoS attack designed to maximize the amount of strain on DDoS defense systems and the collateral damage incurred by defending networks, thereby wreaking havoc on wide swaths of the Internet. We then survey DDoS mitigation solutions in the real world to evaluate the attack against the mitigation solutions. Finally, the chapter demonstrates the feasibility of the attack and quantifies the amount of damage the attack can impose by abusing the mitigation solutions. We thus hope that this chapter can motivate the network community to address the design issues of existing DDoS defense systems and work towards the ideal DDoS defense system we assumed in Chapter IV.

The content in this chapter appeared in the following unpublished work: Shi, L.; Sisodia, D.; Li, J.; Zhang, M.; Dainotti, A.; Reiher, P., The Moving Target Attack: On Quantifying Multi-Wave Victims in DDoS Mitigation. Lumin Shi is the leading author of this work and responsible for leading all the presented analyses.

5.1 Introduction

While many attackers are only interested in attacking specific services on the Internet, some launch DDoS attacks for sport [129, 199]. An attacker with the capability of launching a large-scale DDoS attack *whose goal is to maximize the amount of disruption caused to the Internet*, should not target a single, wellprovisioned service, but instead many under-provisioned edge networks. Such an attack fits the motives of attackers in the latter group above and may have devastating consequences. Therefore, the network community needs to study the attack's consequences and use the results as motivation to improve current DDoS defense mechanisms deployed on the Internet to prevent the attack.

Edge networks that receive attack traffic, defined as **first-wave victims** in this chapter, continue to perform inadequate defense to mitigate link-flooding attacks. They rely on DDoS mitigation protocols such as remotely triggered black hole (RTBH) [97] filtering and BGP FlowSpec [123] to deploy traffic filters at routers/switches in their upstream networks. The upstream networks will then distribute the filters at their traffic ingress points to mitigate unwanted traffic before it propagates within the networks, hoping to unclog the links connected with the edge networks. However, such mitigation protocols above have shortcomings. For example, RTBH removes all traffic, benign or malicious, towards a specified destination network prefix, which can generate massive amounts of collateral damage or the filtering of traffic from legitimate sources, which we define as **second-wave victims**. While BGP FlowSpec enables fine-grained traffic filters with its extensive list of supported IP header fields, many networks implement them in their network routers or switches with little ternary content-addressable memory (TCAM) for storing filters [145]. Hence, networks often limit the number of filters their customer (downstream) networks can deploy, limiting the attack traffic they can cover.

Worse, fine-grained traffic filtering alone does not warrant effective DDoS mitigation. The first-wave victims must accurately identify attack flows before mitigating them. For example, a first-wave victim may filter all DNS responses from a set of DNS resolvers who are exploited by a DNS amplification attack. Unfortunately, not all DDoS attacks are amplification attacks, and the victim is at the mercy of the attacker who decides how to construct each packet. In other words, the network may never derive filters that can effectively mitigate the attack. Therefore, first-wave victims often use the source IP addresses of the attack traffic as a primary factor to match and filter attack traffic. Even then, the source-IP-based filtering method is only feasible when the victim has enough memory capacity on their routers *and* the attack does not employ IP spoofing (i.e., a technique that allows bots to spoof their IP addresses). Sadly, to date, IP spoofing remains a significant problem on the Internet, as shown by Luckie et al. [111].

Given the above observations, we present the moving target (MTA), a link-flooding attack that introduces a **mitigation dilemma** to many first-wave victims: each victim either endures the attack or disconnect itself from a wide spread of second-wave victims during attack mitigation. The second-wave victims include disconnecting networks that originate both attack and benign traffic and ones that do not originate any attack traffic. MTA combines the techniques introduced in prior DDoS attack research (details in Sec. 5.2) and combining them to make itself practical to launch.

In this chapter, we make the following contributions:

- We design MTA, a practical and novel attack that exposes DDoS mitigation issues in the real world quantitatively; this is the first work that studies the consequences of launching a DDoS attack against multiple ASes simultaneously with practical techniques.

– We survey network operators to learn their DDoS mitigation solutions and the limitations of the solutions; the survey results validate the assumptions we make in this draft.

– We then evaluate the attack against real-world mitigation solutions and demonstrate that the attack can inflict severe damage to today's Internet; specifically, we show many first-wave victims have to disconnect themselves from nearly half of the usable IPv4 address space to mitigate the attack.

– We finally summarize two critical countermeasures to build a more robust Internet for defending against MTA.

5.2 Related Work

To launch a link-flooding attack, adversaries take either (1) a direct approach that sends attack traffic (e.g., TCP SYN, UDP, HTTP) from their bots towards the targeted links or (2) a reflective approach that leverages spoofed packets to exploit publicly-accessible services (e.g., DNS/NTP) to amplify the attack bandwidth. These two approaches are the foundation of the advanced attacks below.

5.2.1 Pulsing Attacks:. First appeared in 2003, a pulsing attack [99, 64, 113, 165, 88] is to disconnect a network with periodic, short-lived traffic pulses. These pulses can lead congestion-aware flows to believe in traffic congestion thereby reducing the sending rate of the congestion-aware flows. In addition, because the pulse duration is often short (e.g., 100s of milliseconds), DDoS detection systems may not detect such an attack since their traffic information feed (e.g., NetFlow/IPFIX) is too coarsely grained to spot the attack pulses.

The main challenge of pulsing attacks is to synchronize the traffic pulses among a botnet. For example, a pulsing attack typically requires a pulsing duration of 100s milliseconds, which is virtually impossible to accommodate across thousands, not to mention millions of bots. Indeed, Park et al. [152] show the synchronization difficulty in practice even with the latest pulsing attacks (e.g., CICADAS [88]) that claim higher feasibility. In the moving target attack, we apply pulsing durations that last for seconds than milliseconds; we trade the stealthiness of a pulsing attack for (1) the attack feasibility in practice and (2) the ability to observe the attack's effectiveness.

5.2.2 The Crossfire Attack:.

In the Crossfire attack [86], botnets distribute their attack traffic among public servers at different networks who share a common upstream network link. The Crossfire attack has a more specific goal than the Coremelt attack [195] which has little control over the link it congests. Specifically, each bot establishes low-bandwidth, protocol-complaint flows (e.g., HTTP) with many servers at different ASes. Because there is no single server and AS that receives all the attack traffic, authors claim the attack is challenging for existing DDoS detection solutions. Implicitly, authors assume that transit networks do not perform always-on DDoS detection on their links. The moving target attack assumes networks can detect bot traffic with 100% accuracy, and it only generates protocol-complaint flows when necessary.

5.3 Moving Target Attack

5.3.1 Overview.

This section presents the procedure of the moving target attack (MTA). The attack begins with a reconnaissance of **bots**, **attack-source networks** (i.e., networks that contain bots), and **first-wave victims** (i.e., networks that receive attack traffic). For each bot, MTA estimates its effective traffic sending rate and tests its IP spoofing capability. For each network, MTA surveys its primary network type (e.g., residential networks, content hosting networks) and primary host types of its subnets (e.g., does a subnet contain both servers and clients?).

Then, MTA leverages `traceroute`, Border Gateway Protocol (BGP), and inferred link capacity information to rule out the networks that the botnet cannot overwhelm (details in Sec. 5.3.2.4). For example, suppose an adversary controls a botnet that can deliver 100 Gbps of attack bandwidth. MTA will not attack a network capable of handling 500 Gbps of traffic bandwidth. Once the information above is collected, the adversary is ready to launch MTA against a set of first-wave victims.

During the attack (defined in Sec. 5.3.3), MTA disconnects one or more first-wave victims simultaneously. Meanwhile, MTA observes the effectiveness of each first-wave victim and infers the mitigation method the network uses (if any); we summarize different mitigation methods as mitigation models in Sec. 5.4. For each mitigation model, MTA employs a traffic generation scheme to maximize its attack damage. Based on the attack effectiveness of a first-wave victim, MTA takes one of the following options:

– Stay the course if the existing attack remains effective;

– Switch to a different traffic generation scheme (Sec. 5.3.4) when the existing scheme is ineffective; or

– Stop attacking the network if the network can mitigate all attack generation schemes in MTA successfully (e.g., the network subscribes to a DPS provider for DDoS protection).

As MTA continues, the adversary may find that some first-wave victims are forced to subscribe to DPS providers. In contrast, the remaining victims either endure the attack [1] or bear the collateral damage in mitigating the attack. Figure 29 illustrates an overview of MTA that contains three groups of first-wave victims (i.e., groups are labeled as 1, 2, and 3). An attacker targets one group of first-wave victims at once, and moves their attack target periodically. As a result, all first-wave victims in each group experience frequent disconnections that severely degrade the user experience of real-time applications (e.g., conferencing calls). Should a first-wave victim choose to mitigate the attack without sufficient mitigation resources, it is then forced to block its communication with second-wave victims whose network may or may not participate in the attack; for example, despite the bottom-left residential network in the figure contains no bots, the first-wave victim may have to filter traffic from the residential network to mitigate the attack.

Note that one can automate the entire attack by implementing the steps detailed in this chapter.

5.3.2 Reconnaissance.

This section details the reconnaissance procedure to gather information on bots, attack-source networks, and first-wave victims to carry out an MTA effectively.

5.3.2.1 Estimate Bot Bandwidth.

In MTA, a botnet only attacks networks that it can overwhelm. Therefore, we estimate the total traffic bandwidth the botnet can push to a network; unfortunately, there is no silver bullet to estimate the bandwidth [158]. The effective link bandwidth from a bot to a first-

[1] Note that in practice, a DDoS mitigation response time varies from seconds to hours, as reported by Moura et al. [134]; a first-wave victim may never react to the attack before the attack ends.

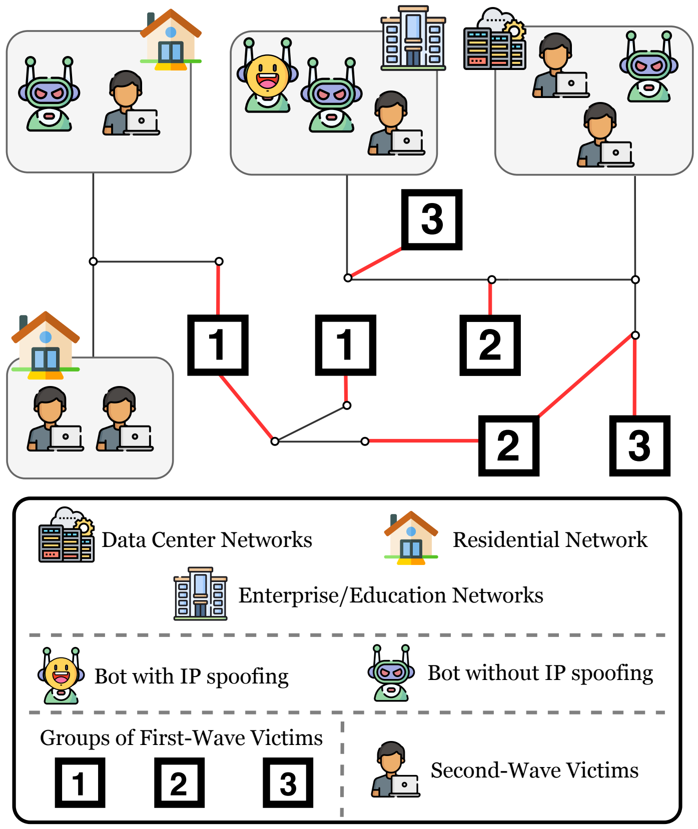


Figure 29. The moving target attack overview

wave victim is dominated by the network hop with the lowest available bandwidth at the time of measurement. While the bot's maximum upload bandwidth can be much higher than the lowest available bandwidth to the first-wave victim, sending more traffic than the link permits would be a waste. Thus, MTA estimates each bot's maximum upload bandwidth per first-wave victim. Specifically, each bot infers the upload bandwidth by transmitting a large request (e.g., HTTP request) to a publicly accessible server of each first-wave victim.

Admittedly, a bot can push more traffic to the first-wave victim than the estimated value, which may cause congestion-aware flows to slow down and land more attack traffic at the first-wave victim. However, other factors such as quality-of-service policies may impose a fixed upload rate on each bot. Thus, such an action brings little benefit and may trigger network operators to investigate the bots. More importantly, a successful DDoS attack does not rely on a few bots.

5.3.2.2 Verify IP Spoofing Capability. IP spoofing allows bots to send attack traffic with arbitrary source IP addresses. It is made possible when stub (edge) networks do not validate the source addresses of traffic leaving the networks. MTA leverages IP spoofing to increase its mitigation difficulty. Therefore, knowing the bots that have spoofing capability is crucial. We provide an elaborated discussion in Sec. 5.6. According to the Spoofer project [22], at the time of this writing, well over 22.5% of $8,067$ autonomous systems (ASes) on the Internet allow IP spoofing consistently. The sampled $8k$ ASes account for less than 10% of all ASes on the Internet. In other words, the actual number of networks that allow IP spoofing can be drastically different. Despite years of research, the Internet continues to allow large-scale IP spoofing to happen [116].

5.3.2.3 Infer Server Subnets. MTA collects the server IPs of each first-wave victim and infers the subnets used for hosting public-facing servers. When necessary, MTA uses the inferred information to generate custom attack traffic for each subnet to increase its mitigation difficulty (details in Sec. 5.3.4 and Sec. 5.4). The subnet host type inference process is as follows. First, an adversary scans a network to gather the host IP addresses with common open ports (e.g., port 80/443 for HTTP/HTTPS). Second, the adversary finds the longest IP prefixes that can cover the host IP addresses; we consider such a prefix to be a subnet of servers. Because network subnets are inherently segmented [156] by host types, the adversary can label the network's subnets based on the dominant host type of each subnet. With high performance network scanners such as *masscan* [63], one can complete a full scan of a /16 network and generate a

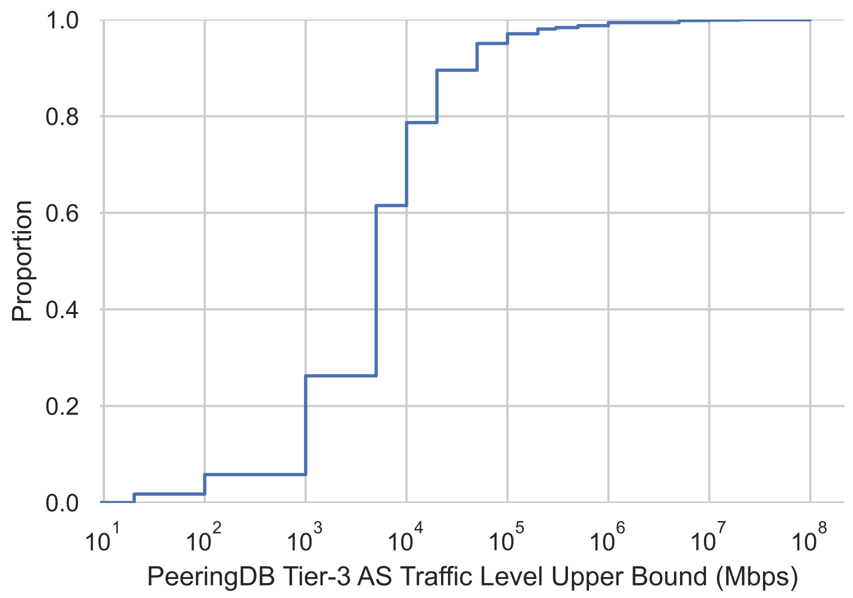


Figure 30. The CDF of tier-3 AS traffic levels.

corresponding report in less than twenty seconds (assuming we only probe 20 ports per IP address).

5.3.2.4 Select Networks to Attack. MTA considers several factors when selecting its first-wave victims. First, the total attack bandwidth of the botnet should be larger than any selected first-wave victim's link capacity. Albeit not always accurate, with online databases such as PeeringDB [155], the adversary can infer a first-wave victim's link capacity and build a list of first-wave victims that the botnet can disconnect. (Figure 30 plots the CDF of the traffic levels of tier-3 ASes)

The adversary may also configure MTA to attack networks that share the same network provider to increase the collateral damage during their attack mitigation (covered in Sec. 5.4). For example, the adversary can perform a traceroute scan to construct the router-level paths from each bot to each first-wave

Table 6. Reported large-scale DDoS attacks in recent years

Attacked Entity	**Year**	**Number of Bots**	**Attack Volume**	**Successful Attack**
DYN	2016	600k	600 Gbps	Yes
Google	2017	Unknown	2.5 Tbps	No (absorbed)
Github	2018	Unknown	1.35 Tbps	Yes
AWS	2020	Unknown	2.3 Tbps	No (absorbed)
Cloudflare	2021	15k	2 Tbps	No (absorbed)
Azure	2021	70k	2.4 Tbps	No (absorbed)
Azure	2022	10k	3.47 Tbps	No (absorbed)

victim. The adversary can then group first-wave victims by their common network providers.

MTA also needs to rule out first-wave victims that major DPS providers protect to avoid wasting attack traffic. Major DPS providers' mitigation capacity ranges from several Tbps to 10s of Tbps [40, 6, 160, 138, 75, 141]. Such a capacity is sufficient to absorb any recent large-scale DDoS attacks, as shown in Table 6. An adversary can employ several approaches to learn DPS-protected networks. First, the adversary can leverage the traceroute results from the botnet to first-wave victims to search for routers' hostnames or IP addresses associated with DPS providers. Second, the adversary may monitor for the round trip time changes of a first-wave victim to infer if anycast is invoked, as proposed by Sommese et al. [190]. Finally, the adversary can monitor the BGP announcements from DPS providers to know the network prefixes protected by DPS providers (i.e., such network prefixes do not belong to DPS providers).

5.3.3 Moving Target Attack Cycle. MTA can disconnect more first-wave victims than a botnet's total attack bandwidth permits. In essence, MTA only send attack traffic towards a group of first-wave victims for a fixed amount of time and then moves on to attack a different group of first-wave victims. An attack cycle is completed whenever MTA begins to attack the first group again. MTA relies on two conditions to change attack targets: *low attack effectiveness* and *pulse duration of an attack*.

The first condition happens when a first-wave victim is under the protection from a DPS provider or the victim's upstream network in rare cases. MTA utilizes bots from different geographical regions as vantage points to observe whether first-wave victims are overwhelmed during attack time, and approaches in Sec. 5.3.2.4 to rule out victims that are just subscribed to DPS providers. The second condition, *pulse duration*, is a property that we extracted from pulsing attacks [99, 64, 113, 165, 88]. In general, a pulsing attack sends short-lived, bursty attack traffic (i.e., pulses) to a network at a frequency. We define the period with attack traffic as *pulse duration*, and the pulse volume as *pulse amplitude*. The attack pulses can force congestion-aware flows to reduce their sending rate (to prevent network throughput collapse). As a result, the pulses can severely impact the user experience of real-time applications such as online gaming and conferencing calls. To address the feasibility issues of the pulsing attacks (discussed in Sec. 5.2), MTA launches attack pulses that last for seconds or a longer period of time.

We illustrate MTA in Figure 31. In each attack cycle, a botnet attacks multiple attack groups. Each group contains a number of first-wave victims, and MTA sends each victim its link-bandwidth amount of attack traffic to disconnect its link. The total pulse amplitude of each attack group should be less than the botnet's total attack bandwidth. For example, MTA completes an attack cycle as follows: The botnet first attacks the attack group that contains AS 1 to AS 4, it then attacks AS 5 to AS 7 in the second attack group. Note that the pulse

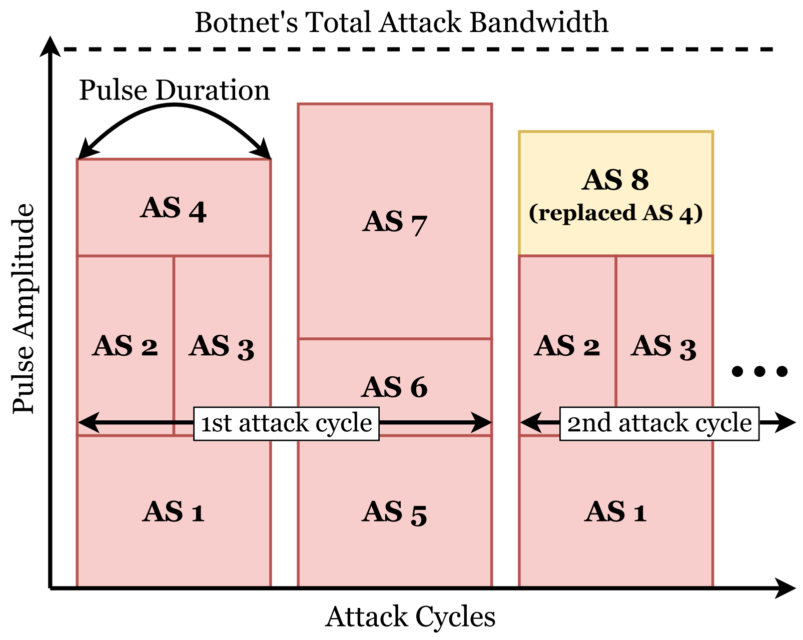


Figure 31. Moving target attack in action

duration does not have to be the same for each first-wave victim (e.g., AS 2 and 3 have a shorter pulse duration than other ASes). In other words, an adversary can increase the number of first-wave victims in each attack group by decreasing the pulse durations of some first-wave victims. Before the botnet attacks group 1 again, MTA replaces AS 4 with another first-wave victim (i.e., AS 8) since the attack effectiveness against AS 4 is low.

5.3.4 Systematic Traffic Generation. The attack traffic generation determines an attack's mitigation difficulty, hence affects the attack's effective duration. MTA observes the attack effectiveness and tailors its attack traffic to each first-wave victims. Below, we introduce three building blocks that MTA employs to generate DDoS traffic.

5.3.4.1 Indiscernible Packets. Attack packets are discernible when they contain shared characteristics that a mitigation system can leverage for accurate traffic filtering (i.e., the mitigation system blocks mostly attack packets and introduces few second-wave victims). Common link-flooding attacks consist of discernible attack packets. For example, in a DNS amplification attack, each amplified packet's source port is 53 and has a DNS resolver's IP address. In such an attack, a network can drop all traffic sourced from port 53 to mitigate the attack and only allow traffic from trusted DNS resolvers.

If a mitigation system cannot find shared characteristics of the attack packets, its mitigation accuracy suffers. Suppose an attack generates packets with random packet headers (e.g., IP addresses, ports) and payloads. The mitigation system cannot mitigate the packets with filters and may introduce a huge number of second-wave victims to benign traffic if it deploys too many filters that happen to cover benign traffic. However, random packet generation is often not a good attack strategy if the first-wave victim has preventative filters deployed at its upstream network(s). For example, the victim may ask its upstream networks(s) to only forward traffic towards a few ports of its subnets, which prevents most randomly generated packets from reaching the victim.

When necessary, MTA generates its attack packets *selectively* (details in Sec. 5.6). It uses the collected server subnet information to generate packets that a first-wave victim has to accept (e.g., the generated packets' destination ports are the common ports of the subnet). As a result, the first-wave victim cannot mitigate the MTA's attack packets using the mitigation approach for common link-flooding attacks.

5.3.4.2 Traffic Dispersion. MTA disperses its attack packets among a set of destination IP addresses of a first-wave victim. (Such a technique is also referred to as carpet bombing [30] by the network community.) Traffic dispersion is the primary technique that enables the Crossfire attack [86], which aims to overwhelm a target network's upstream link(s). However, the target network only receives part of the attack traffic. Hence, the authors of the Crossfire attack claim such an attack is difficult to detect. Unlike the Crossfire attack, we assume first-wave victims can always detect all attack traffic with perfect accuracy, and we focus on evaluating the mitigation dilemma a first-wave victim faces.

In case the network employs a stateful defense method, MTA disperses its attack traffic among the subnets that host services open to the Internet to increase the amount of attack traffic the network receives (see Sec. 5.6.6). Bots may also apply the traffic dispersion technique to its source IP addresses if they can spoof IP addresses.

In contrast, common DDoS attacks target one or a few IP addresses of the first-wave victim. Coarsely-grained filtering techniques, such as RTBH, are often sufficient to mitigate the attack with manageable collateral damage; RTBH removes all traffic (benign or malicious) towards the attacked IP addresses. (details in Sec. 5.4). In other words, such coarsely-grained mitigation is undesirable against traffic dispersion.

5.3.4.3 Stateful and Stateless Attack Traffic. Depending on the mitigation method of a first-wave victim, MTA generates attack traffic in either stateless or stateful mode. By default, MTA runs in the stateless mode where it sends TCP SYN or UDP packets with port numbers that the first-wave victim allows, and MTA discards the responses from the first-wave victim. However, in case the first-wave victim employs stateful mitigation solutions that filter traffic that does not comply with traffic protocols (e.g., TCP handshake), MTA will follow the protocol to bypass the mitigation. From there, MTA can generate traffic at a **congestion-unfriendly** rate to cause link congestion.

5.4 DDoS Mitigation Primer

In this section, we cover (1) where link-flooding DDoS mitigation takes place in practice, (2) how real-world mitigation solutions defend against link-flooding attacks, and (3) a deployment challenge of DDoS mitigation. We also summarized the practical mitigation solutions based on their traffic filtering granularity and implementation location in Table 7.

5.4.1 Where to Mitigate Attacks. If a first-wave victim's link is fully occupied by attack traffic, it is too late for the first-wave victim to mitigate the traffic locally. The victim needs to mitigate the attack traffic at the link's upstream locations to unclog the link. If the first-wave victim is multi-homed, it needs to mitigate attack traffic at multiple upstream networks for an effective defense. Unfortunately, the response time from an AS receiving the mitigation request to the AS starting the mitigation varies from one AS to another. Remote upstream ASes that are closer to the attack-source networks are unlikely to offer mitigation quickly, which we consider as an unsuccessful mitigation.

To address the issues above, DPS providers can help first-wave victims to absorb/cleanse their attack traffic. It requires a DPS provider, who agrees to mitigate the attack of a first-wave victim, to become a remote upstream AS that carries most of the first-wave victim's traffic. Specifically, the DPS provider, with multiple geographical points of presence (PoP), announces border gateway protocol (BGP) messages to its neighboring networks saying that it can deliver traffic to the first-wave victim with better metrics (e.g., short AS path, specific prefixes, or pre-established policies). In general, more traffic of the first-wave victim will traverse through the DPS provider [135, 173], if the provider has a large number

Table 7. Common DDoS mitigation solutions.

Mitigation Solution	Source IP/Prefix	Destination IP/Prefix	Other Header Fields	Payload	Filtering Granularity	Common Implementation
Stateless Solutions:						
RTBH		X			Coarse	Hardware
S/RTBH	X				Coarse	Hardware
BGP FlowSpec	X	X	X		Medium	Hardware
BPF w/ XDP	X	X	X	X	Fine	Software
Stateful Solutions:						
Conntrack	X	X	X		Fine	Software
App-specific mitigation	X	X	X	X	Fine	Software

of geographically diverse PoPs. Today, major DPS providers have 100s of PoPs at Internet exchange points (IXP) in different countries.

5.4.2 How to Mitigate Attacks.

DDoS defense requires (1) a detection system to tell the attack and benign traffic apart and (2) a mitigation system that removes the attack traffic. A defense solution is either stateful or stateless.

Stateful defense: A stateful defense's detection system and mitigation system are tightly coupled. The mitigation system tracks every packet to update the corresponding flow statistics and decides if a packet should be dropped based on criteria such as protocol state at transport or application layers. Such a solution is deployed **in line** with production traffic where both the detection and mitigation systems will affect the network throughput; the solution has a short time budget to process each packet. Due to the constrained budget, they are best suited for ruling out traffic that is obviously malicious. For example, `conntrack` (a transport-layer stateful firewall) tracks the status of each TCP connection bidirectionally and drops packets with invalid states in real-time. The firewall creates a flow entry for each TCP SYN packet it sees. It then waits for a valid SYN-ACK and an ACK packet to ensure the TCP flow is properly established. By default, the firewall drops all other TCP packets unless their corresponding flows are tracked and established. Meanwhile, a layer-5 (e.g., web server) defense solution may have arbitrary policies to govern the requesting rate of each user.

A stateful solution tends to fail to mitigate link-flooding attacks due to the constrained computational and I/O resources. DPS providers alleviate the resource constraint by processing traffic at multiple data centers so that each server only handles a subset of the traffic of a first-wave victim. Since stateful mitigation solutions only mitigate attack traffic that does not play by their rules, they do not work when bots play the rules. E.g., a bot can establish valid TCP connections to bypass `conntrack`.

Stateless defense: A stateless defense solution uses an out-of-band detection system. Since the detection solution does not affect the production traffic, it has a copious time budget to detect attack traffic. Often, the detection system also relies on domain experts to decide the proper detection thresholds. For the sake of simplicity, in this chapter, we assume such a detection system produces results with perfect accuracy (no false positives or negatives). We also assume detection solutions can provide the mitigation system a list of 5-tuple attack flows[2]. Then, the mitigation system matches attack flows and applies the corresponding action (e.g., drop, rate limit). Despite the perfect detection results, a stateless mitigation solution will generate second-wave victims. For example, the direct upstream AS may only provide solutions such as RTBH that can only match and filter traffic by destination IP addresses, which essentially disconnects a first-wave victim from

[2]A 5-tuple often identifies a flow with source/destination IP addresses, source/destination ports, and its transport-layer protocol.

the Internet. Fine-grained solutions such as BGP FlowSpec often come with a constrained filter capacity that is insufficient to filter attack flows by individual IP addresses.

5.4.3 A Deployment Challenge: Bidirectional Traffic.

While first-wave victims can see traffic bidirectionally, they cannot use the stateful defense to mitigate the attack locally. For a stateful defense solution to function, it requires the bidirectional traffic of a network flow. However, Internet routing is asymmetric, so the upstream ASes of the first-wave victim may only see the network traffic towards the victim, but not the other way around. Suppose the first-wave victim wants to enable a stateful defense solution at its upstream ASes who can unclog its link. The victim needs to ensure that the outbound traffic of a network flow is sent to the upstream AS that forwards the inbound traffic of the flow. Each upstream AS of the victim then needs to ensure the bidirectional traffic of each flow is processed by the same device that hosts the stateful defense [3]. Unfortunately, the above operation is extremely challenging to coordinate among multiple upstream ASes that carry the first-wave victim's traffic. We have yet to see large-scale deployment of stateful defense conducted by transit ASes.

Note that without the bidirectional traffic, the DDoS detection system of a stateless defense can only classify significant outliers (e.g., an IP sends an excessive number of SYN packets). The stateless mitigation can continue to function since it is not coupled with any detection systems.

5.5 A DDoS Mitigation Survey

We surveyed the North American Network Operators' Group (NANOG) about the mitigation solutions network operators use and how widespread the DPS

[3]One may implement symmetric hashing [36] to achieve such a setup.

providers are among the networks. Appendix A.1 includes a link to our survey questions and responses. We received a total of *27* valid responses. Below, we summarize the most relevant survey statistics from our survey result.

Our survey respondents are from different tiers of networks. Three are from tier-1 networks (e.g., CenturyLink), eight are from tier-2 networks (e.g., Hurricane Electric), and fifteen are from tier-3 networks (e.g., educational networks, online gaming). While most respondents (20 out of 27) treat link-flooding attacks as a major problem in their networks, seven respondents do not treat it as a significant problem. Based on the answers of seven networks, **we find they either never experienced large-scale link-flooding attacks or experienced attacks that only last for a minute on average**. Nineteen respondents (70% of all correspondents) encounter link-flooding attacks at least once per month. Twelve of them experience link-flooding attacks once per week or more frequently.

We find RTBH (a high collateral damage solution) is widely employed by the respondents' networks; 24 out of 27 networks (85%) use RTBH as one of their mitigation methods. The remaining three networks rely on fine-grained mitigation or DPS protection. Out of the three networks, the first respondent never encountered a link-flooding attack that is large enough to cause any issue, the second respondent chooses to filter traffic in the local network, meaning that the respondent may misunderstand our survey, and the last respondent uses a custom signaling protocol to deploy traffic filters at upstream networks. Among the 24 networks that employ RTBH, **seven** only use RTBH as their mitigation method against link-flooding attacks, **eight** use BGP FlowSpec or custom signaling protocols offered by their upstream networks, and the remaining nine networks subscribe to DPS providers, which is **a one-third of the 27 networks**.

Among the eight networks that employ fine-grained mitigation capability, three states that their upstream networks limit the number of filters they can deploy. The filter limit ranges from **tens** to **ten-thousands** of filters; such a filter limit is not surprising given the popular switching platforms only support thousands of filters [145]. Four of them do not know the number of filters their upstream network support. Only one network claims that its upstream network(s) do not have a filter limit.

This chapter assumes networks already subscribed to DPS providers are free from MTA due to the sheer amount of computational and network resources available at geographically distributed data centers. With that said, we believe a separate study is required to estimate a DPS's actual filtering capacity. For example, the research could study how effective a DPS provider's BGP anycast is and how does the filter complexity affect a DPS provider's network throughput. Instead, MTA focuses on networks that do not subscribe to DPS providers and rely on their upstream networks for attack mitigation (i.e., our survey has 18 (66.6% of all respondents) networks).

Given the survey results and the stringent requirement of stateful mitigation solutions, we realize a significant number of networks rely on stateless mitigation solutions to defend against link-flooding attacks. These solutions can introduce high collateral damage in mitigating DDoS attacks, as shown in Sec. 5.7.

5.6 Mitigation Models Against MTA

Based on the analysis of DDoS defense solutions and the survey results in Sec. 5.4 and Sec. 5.5, we use three mitigation models (M1, M2, M3) to cover the mitigation methods in practice. For each mitigation model, we cover (1) the suitable MTA strategy against it, and (2) the conceivable collateral damage caused by the model with and without IP spoofing.

5.6.1 Stateless Coarse-Grained Mitigation (M1). M1 employs RTBH and source-based RTBH (S/RTBH) to mitigate attack traffic by either destination or source IP addresses, respectively; they cannot take advantage of fine-grained DDoS detection results (e.g., attack flows presented in a 5-tuple format). Specifically, RTBH drops all traffic towards the IP addresses of a first-wave victim. If the RTBH is deployed by the *direct* upstream ASes of the first-wave victim, they effectively disconnect the first-wave victim from the Internet. On the other hand, if the first-wave victim can deploy RTBH filters at remote upstream ASes that are close to or at the attack-source networks, the collateral damage is reduced; it no longer blocks all networks from accessing the IP addresses. Unfortunately, Nawrocki et al. [137] show the latter scenario rarely happens in practice — remote ASes rarely accept inter-AS RTBH messages.

With S/RTBH, an AS can drop all traffic from the IP addresses that generate attack traffic. Because S/RTBH only drops traffic by source IP addresses, when deployed, the AS blocks all their customers networks (including the first-wave victim) from accessing the IP addresses that generate attack traffic. Therefore, S/RTBH is best deployed at locations close to the first-wave victim so the victim's neighboring customer networks remain unaffected. For example, if an upstream AS deploys S/RTBH filters at the egress port that directly connects to the first-wave victim, all other customer networks are not affected. However, in a link-flooding attack, it is too late to deploy a filter at the egress port, and the best practice is to filter attack traffic at its ingress points, which contradicts the example above and renders the method rarely practiced (this is also reflected by Sec. 5.5).

5.6.2 MTA Against M1.

Since M1 cannot match and filter traffic based on layer-4 information, MTA does not need to craft traffic defined in Sec. 5.3.4.1. In other words, each bot can simply blast the same packet at a first-wave victim that employs M1. Instead, MTA employs the traffic dispersion technique (Sec. 5.3.4.2) to distribute the attack traffic among all IPs of the first-wave victim, which ensures that the victim has to include take all its IPs offline to mitigate the attack. In the unusual case where S/RTBH is enabled, MTA with IP spoofing can cause the network provider to disable all its clients from reaching the Internet. MTA detects the usage of M1 as follows:

– **RTBH**: Ask geographically distributed vantage points to connect with the IP addresses under attack. RTBH is employed when all such attempts fail to establish.

– **S/RTBH**: The adversary checks whether his/her bots can reach the first-wave victim's sibling networks (i.e., networks that share the same provider). S/RTBH is employed when none of the bots can communicate with the sibling networks.

5.6.3 Stateless Fine-Grained Mitigation (M2).

M2 includes but is not limited to BGP FlowSpec or BPF with eXpress Data Path (XDP) [69]. They offer fine-grained traffic mitigation, which can introduce fewer second-wave victims than M1. For example, a first-wave victim can ask its upstream network to forward only HTTP(s) traffic to one of its subnets. BGP FlowSpec is a filter dissemination protocol that allows its users to filter traffic by layer 3 and 4 packet header values. Its filters are often implemented in hardware routers that rely on low-latency memory (i.e., CAM/TCAM) for packet matching and filtering. BPF with XDP is an efficient software-based filtering solution. Specifically, XDP allows BPF to filter packets in kernel space before the kernel's network stack constructs the packets.

While each filter in M2 generates virtually no second-wave victims, M2 is constrained by its filtering capacity, which forces it to use a limited number of filters to mitigate DDoS attacks. For example, a first-wave victim may only able to deploy one thousand filters at its upstream ASes, which prevents the victim from mitigating an attack that involves thousands of bots. To mitigate the attack, the victim needs to use network prefixes to cover as many bot IPs as possible.

5.6.4 MTA Against M2. A first-wave victim may leverage M2 to employ strict network policies to prevent unsolicited traffic from congesting its network link. MTA generates attack packets selectively as described in Sec. 5.3.4.1 and disperses the attack packets as described in Sec. 5.3.4.2. The first technique allows MTA to bypass the network policies (if any) to ensure its traffic can reach the first-wave victim. The latter technique forces the first-wave victim to deploy more filters or use coarse-granular filters. The problem worsens when MTA employs IP spoofing: A first-wave victim can disconnect itself from the Internet with automated mitigation under the limited filter capacity.

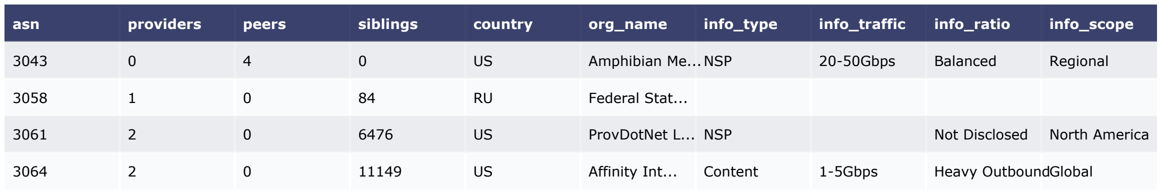
MTA can detect M2 with the help of vantage points that are not part of the attack. Specifically, M2 is engaged if the vantage points can establish connections with the first-wave victim IPs during the attack.

5.6.5 Stateful and Stateless Mitigation (M3). Stateful mitigation solutions are deployed in-line with the production traffic. They rule out traffic that does not follow transport or application protocols with little to no collateral damage. For example, `conntrack` drops TCP packets with incorrect states (e.g., sequence numbers). A web application firewall (WAF) may use a reCAPTCHA-like system to prevent bots from reaching first-wave victims directly. Note that bots may still bypass reCAPTCHA if they have ample computational resources. These solutions face performance issues at upstream networks to withstand large-scale attacks; they require immense computational and I/O resources. Indeed, during a large-scale attack, we often see under-provisioned stateful solutions (43% reported DDoS attacks) to cause a denial of service on the protected networks [139].

A network can combine both stateless and stateful mitigation solutions to remove attack traffic. For example, the network may first use a stateless mitigation solution to remove obvious attack traffic and have the stateful mitigation solution to handle the remaining traffic. However, only DPS providers have the resources to deploy stateful solutions to absorb large-scale link-flooding attacks in practice.

5.6.6 MTA Against M3. The attack strategy against M3 is identical to the strategy against M2 (Sec. 5.6.4), and the explanation is as follows. An attack can involve 10s or even 100s of thousands of bots; these bots force M3 to track a devastating amount of flows. Mian et al. [128] and Corin et al. [41] show that the throughput of a software-switched network is severely reduced with only *thousands* of traffic filters. Depending on the packet volume, packet rate, and how packets are processed in software, the throughput reduction ranges from 30% to 80%.

Furthermore, with IP spoofing, MTA can render stateful mitigation solutions fruitless: IP spoofing not only allows bots to spoof addresses but to **bypass stateful firewalls for an extended period**. Specifically, a stateful firewall must allow the very first packet of a connection (i.e., SYN packet of a TCP connection) to reach its destination IP address. Only then can the firewall drop the same subsequent packets from the packet source. In other words, should each bot only



asn	providers	peers	siblings	country	org_name	info_type	info_traffic	info_ratio	info_scope
3043	0	4	0	US	Amphibian Me...	NSP	20-50Gbps	Balanced	Regional
3058	1	0	84	RU	Federal Stat...				
3061	2	0	6476	US	ProvDotNet L...	NSP		Not Disclosed	North America
3064	2	0	11149	US	Affinity Int...	Content	1-5Gbps	Heavy Outbound	Global

Figure 32. A snapshot of the AS information database (with a reduced number of columns).

sends *unique* SYN packets during the entire attack period, the firewall will forward all the attack packets. We show adversaries can launch such an attack indefinitely; the calculation of the effective attack length is provided in Appendix A.3.

5.7 Evaluation

5.7.1 Overview.

The first-wave victims of MTA: Since the point of MTA is to put *multiple* first-wave victims in a mitigation dilemma, we study how many first-wave victims an adversary can disconnect simultaneously given different *magnitudes of attack power* and *attack scenarios.* (Note that the main first-wave victims are tier-3 ASes; they *source* information but rely on network links that ordinary DDoS attacks can overwhelm.) We approach the problem from three aspects as follows: **First**, we demonstrate how to infer the ASes that are protected by DPS providers; we developed a tool to track if an AS is protected by a DPS provider in near real time. MTA can rely on this tool to determine whether it should spend its attack power on the AS or not. **Second**, we use collected network information to infer the link capacity of ASes. A high inference accuracy allows MTA to better estimate the required attack bandwidth to overwhelm a group of first-wave victims. While an attacker can target a first-wave victim with a gradually increasing attack volume to estimate the required amount of attack traffic of each first-wave victim more accurately, the attacker can leverage the link capacity inference to reduce reconnaissance time. **Lastly**, with the bandwidth inference result, we show the relationship between the magnitudes of attack power and the first-wave victims an adversary can disconnect. Specifically, we show the various types of possible victims with and without the moving target attack technique.

The second-wave victims of MTA: Each first-wave victim is in a mitigation dilemma — it either: (1) does not mitigate the attack and suffer a poor network goodput or (2) mitigates the attack and (un)intentionally introduces second-wave victims (i.e., the IP addresses and ASes who do not carry any DDoS traffic). To make an informed decision, a first-wave victim needs to quantify the second-wave victims given its mitigation model.

For obvious ethical and legal reasons, we cannot launch real DDoS attacks against networks on the Internet. Therefore, we built a simulation to examine the scale of the second-wave victims under different magnitudes of attack power and mitigation resources. The simulation leverages real-world network measurement data and our DDoS mitigation survey results to synthesize the botnets and each first-wave victim's defense model.

5.7.2 The First-Wave Victims of MTA.

5.7.2.1 Experiment Setup. This section introduces our primary datasets and how we processed them to facilitate our understanding of the potential first-wave victims. First, we use the CAIDA AS relationship dataset [25] to find tier-3 ASes (i.e., ASes with no customer networks) on the Internet. For each tier-3 AS, we maintain counters to track its providers, peers, and sibling ASes. We then augment each AS number (ASN) with CAIDA prefix to ASN [23], CAIDA AS organizations [24], and PeeringDB datasets [155]. As a result, we built an AS information database, as demonstrated in Figure 32. Each row is identified by an ASN and contains the information about the AS (e.g., neighbor AS counters, organization name, geographic location, traffic level, network type, and its network prefixes). Second, we collect and parse BGP updates from a total of 61 BGP route collectors from different geographic locations (using BGPKIT [16]) and monitor for the BGP announcements originated from major DPS providers [59]. We then manually select the corresponding ASNs of the major DPS providers using the AS information database above.

5.7.2.2 The Subscribers of DPS Providers. As part of the MTA design, it does not waste its attack power on ASes with over-provisioned resources, (i.e., the ASes who pay for anycast-based DDoS defense offered by major DPS providers), we developed a tool to conduct a week-long BGP-based measurement study to find such ASes. In other words, the study is to find the ASes that were under DDoS attacks and then protected by DPS providers (as listed in A.2). Note that anyone can use the tool and live BGP updates to track ASes under DPS protection in near real time.

Figure 33 applies the cumulative distribution function (CDF) to the number of DPS subscribers in a seven-day time window. It indicates how busy the DPS providers are given a time frame; we find the DPS providers initiated DDoS mitigation efforts to protect 46 ASes within a seven-day time window. Note, in the first day, we captured over 50% of all the protected ASes in the entire time window. Subsequently, we captured only 3 to 4 newly protected ASes each day. This implies that there is a limited set of ASes that are constantly protected by DPS providers.

5.7.2.3 The Accuracy of Link Capacity Inference. To know who are the potential first-wave victims in MTA, we infer the link bandwidth of the ASes on the Internet and find the ASes whose links can be fully disconnected by

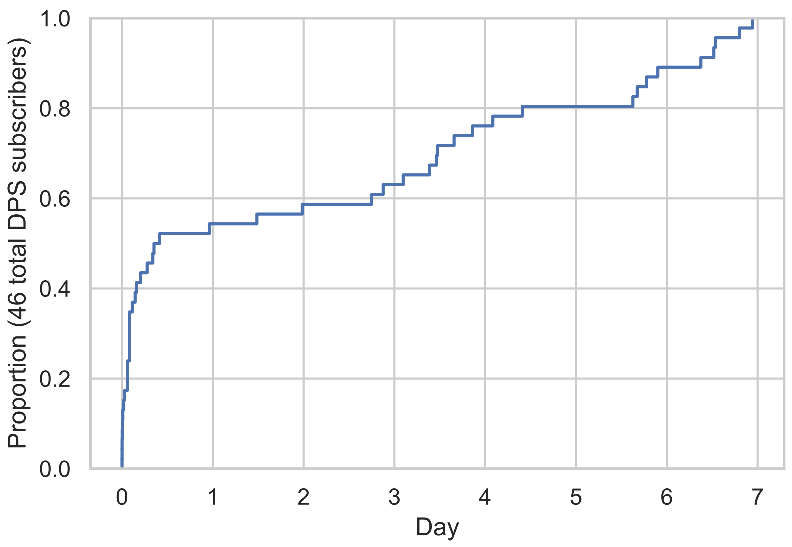


Figure 33. The CDF of the number of DPS subscribers.

a botnet. At the time of this writing, 7,239 tier-3 networks disclosed their traffic level information on PeeringDB while there are more than 61,300 tier-3 networks on the Internet. In this chapter, we assume the traffic level of a network is 50% of the network's link bandwidth. E.g., if a network's traffic level is 5Gbps, then we assume its link capacity is 10Gbps. The assumption is a common convention accepted by the network community [67].

Because we do not have the resources to manually ask for each of the remaining networks about its network capacity, we infer each network's traffic level instead. We used *scikit-learn* [20] and applied three classification methods: *decision tree*, *k-nearest neighbors (KNN)*, and *random forest*, on the features available in our AS information database to infer the network capacity of each AS. Specifically, the selected AS features are *ASN*, *provider count*, *sibling count*, *peer count*, and *IP count*. We reduced 16 traffic levels available in PeeringDB to 4 traffic levels for reasons explained in Appendix A.4. To train and test each classifier, we used a

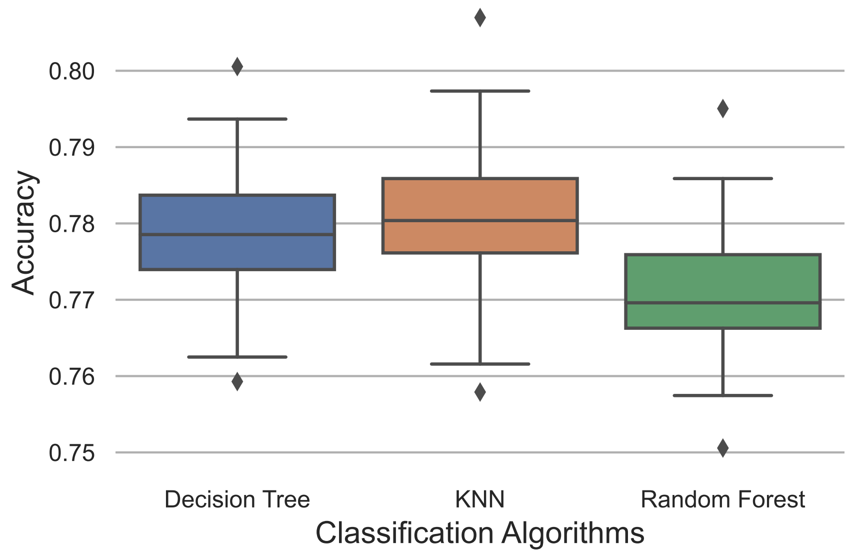


Figure 34. The traffic level inference accuracy of three classification algorithms.

70%/30% data split for training and testing. We then evaluate each classifier 100 times with random states.

Figure 34 shows that the three classifiers can infer traffic levels with at least 77% median accuracy. Out of the three classification algorithms, KNN provides the best overall accuracy on our dataset. We use the trained KNN model to infer each AS's link capacity for the remaining studies. Note that the inference accuracy can vary significantly as networks update their traffic levels on PeeringDB. For example, in 2018, Smith et al. [186] reported a 90% inference accuracy of link capacity of transit ASes using decision tree. The discrepancies in inference accuracies is due to the updated information in PeeringDB and the type of ASes that we are evaluating.

5.7.2.4 Attack Power vs. First-Wave Victims. With the link capacity inference results, we can now study the number of first-wave victims an adversary can introduce given different magnitudes of attack power with MTA.

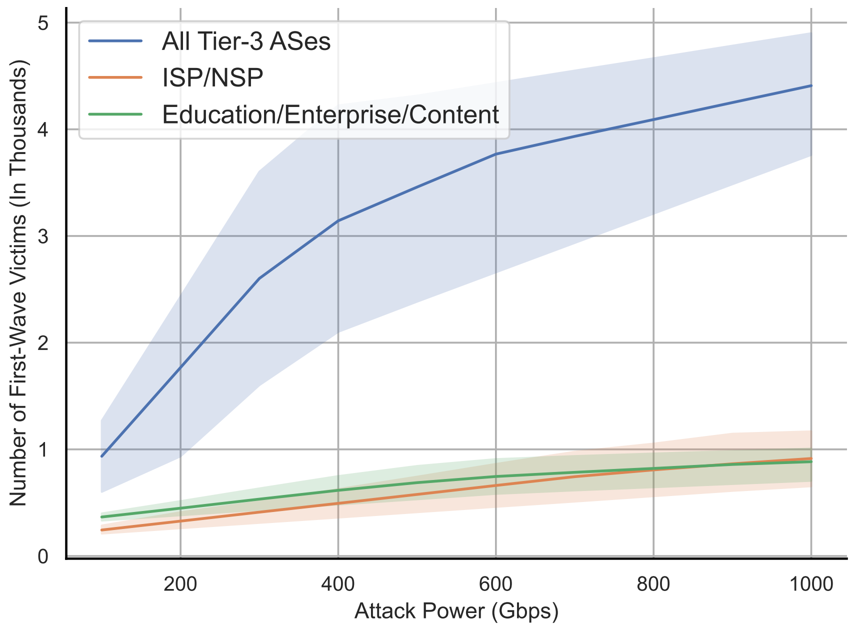


Figure 35. Estimated first-wave victim ASes.

To maximize the number of first-wave victims, we first rank all tier-3 ASes by their link capacity from low to high, and then apply MTA against the ASes from the lowest end. For each first-wave victim, we use attack traffic that is worth 150% of the victim's link capacity to disconnect it. Because we cannot launch actual DDoS attacks to measure the required attack bandwidth for each first-wave victim, we choose a constant factor (i.e., 150%) to compensate for the attack traffic transmission loss due to various network conditions such as early link congestions. The study contains four MTA profiles that ranges from one attack group to four attack groups. When MTA only has one attack group, MTA focuses all its attack power on the same set of first-wave victims. Meanwhile, when MTA has four attack groups, MTA rotates its attack power among 4 sets of first-wave victims, which introduces more first-wave victims.

Figure 35 illustrates the numbers of potential first-wave victims by their network types. Specifically, the types are (1) tier-3 ASes, (2) Internet/network

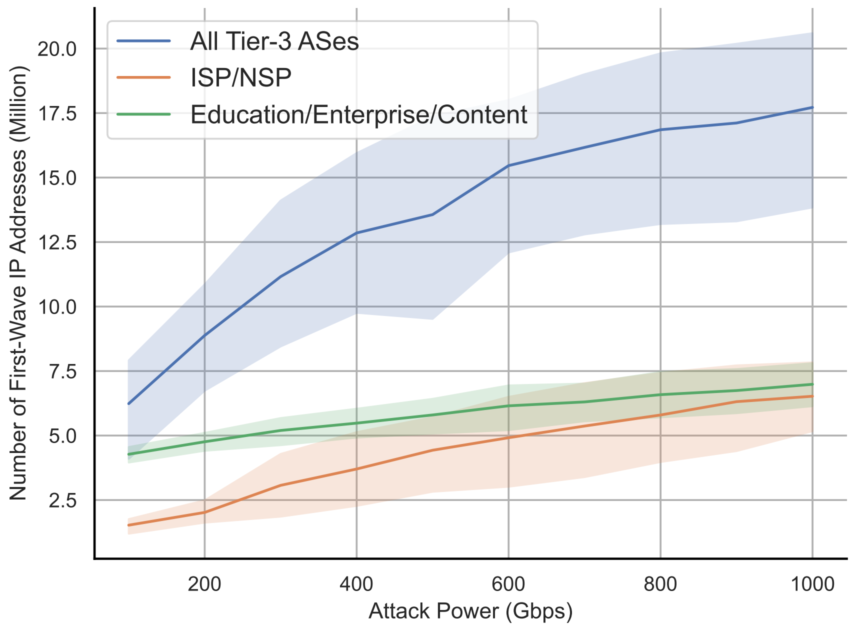


Figure 36. Estimated first-wave victim IP addresses.

service providers (ISP/NSP), and (3) education/enterprise/content networks. Due to the limitation of PeeringDB dataset, only a subset of ASes disclosed their network types. Therefore, we expect the results of the latter two types to change as more ASes discloses their network types. With just 100 Gbps of attack throughput and MTA that attacks four groups of ASes, an adversary can attack up to 295 ISP/NSP ASes. Worse, the adversary can disconnect over 1,436 first-wave victims when he/she attacks any tier-3 ASes. When an adversary poses a botnet that can deliver 1Tbps attack throughput, it can periodically disconnect over 5,000 ($\approx$ 8.2%) tier-3 ASes with MTA that attacks four groups of tier-3 ASes. This study is an estimation of the potential number of first-wave victims. It does not consider the mitigation strategies each victim applies. We consider mitigation models in the evaluation of second-wave victims.

Figure 36 shows the number of first-wave victim IP addresses affected by the same attacks above. For example, with the 100 Gbps attack above, the 295

ISP/NSP first-wave victims own a total of 1.6M IP addresses. If we map each IP address to a household, the 100 Gbps DDoS attack can cause 1.6M homes to experience frequent Internet disconnection when the first-wave victims do not react to the attack. To put the number into perspective, such an attack could potentially disconnect more households than the Los Angeles city (1.38 million households) according to the U.S. Census Bureau [29]. With the same attack throughput, the attacker can disconnect up to 7,504,129 IP addresses when attacking any tier-3 ASes.

5.7.3 The Second-Wave Victims of MTA.

5.7.3.1 Experiment Setup. The experiment is a simulation-based study that evaluates the second-wave victims caused by the first-wave victims who employ M2 under various attack scenarios. (Note that the attack strategy against M3 is the same as M2.) It is unnecessary to evaluate ASes that employ M1 (Sec. 5.6.1): Such an AS blocks all incoming traffic from all other ASes on the Internet; the second-wave victims are the entire Internet to the AS.

Because M2 employs fine-grained filters that matches source IPs of the bots in an attack, the source IP distribution affects what filters are generated hence critical to the experiment. Therefore, we built a set of synthesized Mirai botnets based on a real-world DDoS incident report [8]. Specifically, given a botnet size and the country distribution of the botnet, we map a set of bots to each country proportionally. Each bot is then assigned with an IP address of its belonging country and its uplink bandwidth using the Ookla's speedtest dataset [148]. The sizes of the synthesized botnets range from 5K to 100K.

We assume each first-wave victim has a perfect DDoS detection accuracy which puts MTA in a disadvantageous situation. With the perfect detection

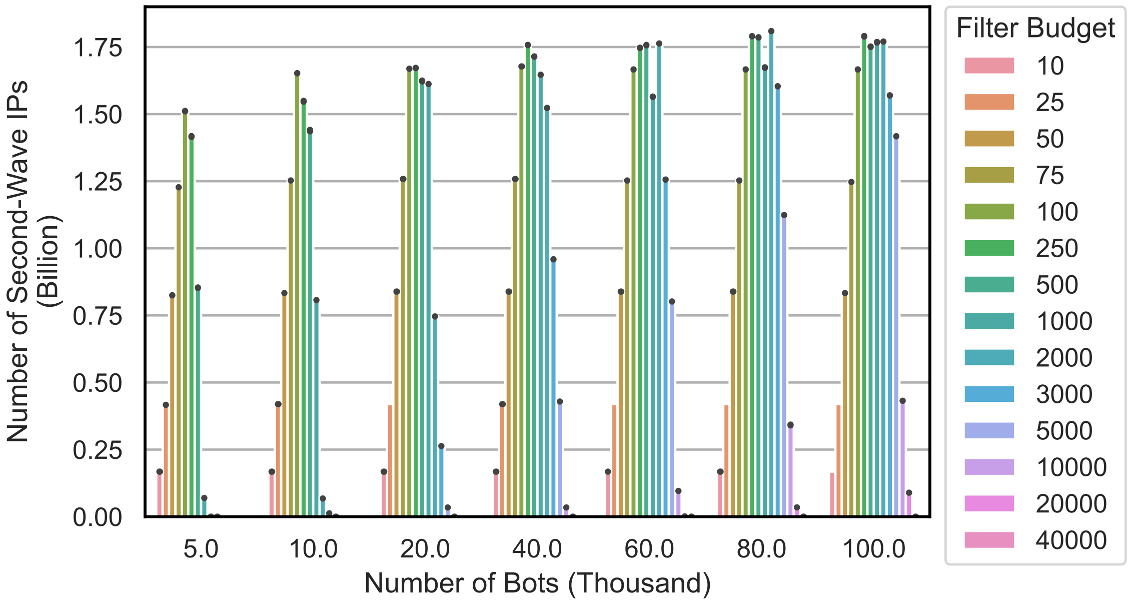


Figure 37. The second-wave victim IP addresses under varying botnet sizes and filter budgets.

result, each first-wave victim also employs a filter generation process as defined in Appendix A.5 to optimize for the attack traffic coverage within a fixed budget. In this study, the shortest filter prefix length the filter generation process can generate is /8, each first-wave victim in the simulation allows attack traffic occupying at most 30% of its link capacity, and for the sake of simplicity, we do not allow bots to spoof IP addresses to increase the mitigation difficulty.

5.7.3.2 Botnets vs. Second-Wave Victims. Figure 37 shows the numbers of second-wave victim IPs under different attacks and filter budgets (i.e., how many fine-grained filters a first-wave AS can deploy). In all attacks, we first see the number of second-wave victim IPs gradually increases as we increase the filter budgets initially. This is because the available filter budgets are too small for mitigating the attacks even we filter attack traffic using /8 prefix filters. Using the 5K-bot attack as an example, as we continue to increase the filter budget, the

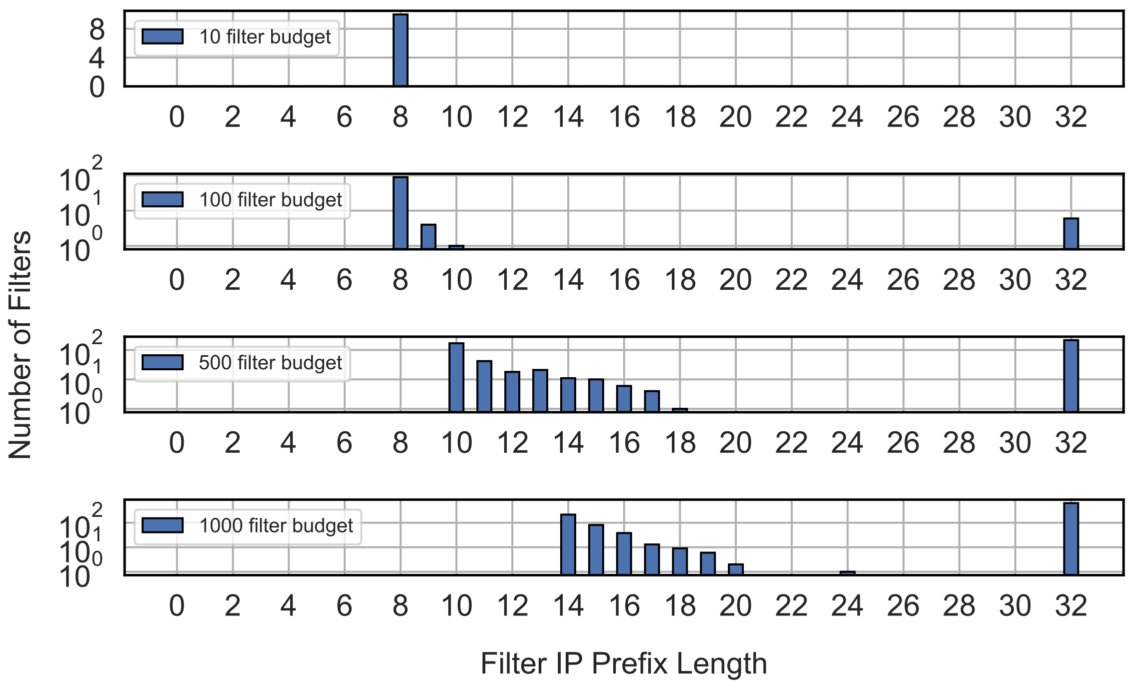


Figure 38. The distribution of filter sizes with different filter budgets and 5K bots.

number of second-wave victim IPs reaches 1.6 billion, which corresponds to 43.2% of the public IP space or 34K to 42K of second-wave victim ASes, at its peak with 100 filters. The affected number of IPs then starts to decline as the filter budget continues to grow. With a 1K filter budget (20% of the botnet size), the number of second-wave victim IPs is reduced to $\approx$ 70M, which corresponds to approximately 3K second-wave victim ASes.

Figure 38 demonstrates how the filter budgets affects the prefix lengths of generated filters under the same 5K-bot attack. We see that the filters become more specific as we increase the filter budget. With 2K filters, we no longer produce second-wave victims, this is because the rule generation process allows attack traffic to occupy at most 30% of an AS's link capacity. In other words, all filters are generated at /32-level (individual IPv4 addresses).

5.7.4 Evaluation Summary.

In this section, we demonstrate the feasibility in choosing first-wave victims from two aspects: First, to avoid attacking networks that are subscribed to DPS providers, we created and evaluated a tool that allows an attacker to track ASes under DPS protection in near real time. We found that over a seven-day period, there is a small set of ASes under constant DPS protection. Second, to increase the probability that each set of first-wave victims can be fully disconnected, an attacker may attempt to infer link capacity of the potential first-wave victims. We proposed an inference method that applies classification techniques to publicly available data from PeeringDB. We show that an attacker can infer traffic levels with at least 77% median accuracy.

We then analyzed the number of first-wave victims an attacker may disconnect given different magnitudes of attack power, and found that with merely 100 Gbps of attack power attacking a total of four sets of first-wave victims, an attacker could disconnect up to 295 ISPs/NSPs (which total 1.6M IPs) or 1,436 tier-3 ASes (which total 7.5M IPs).

Lastly, we analyzed the impact that deploying stateless fine-grained filters (M2) has on second-wave victims. We show that the number of second-wave victims in fact (counterintuitively) increases as the filter budget increases to a certain point, and decreases after that point. For example, with a 5K-bot and budgets of 10 filters, 100 filters, and 1000 filters, the number of second-wave victims are 167M IPs, 1.5B IPs, and 70M IPs, respectively.

5.8 Countermeasures

5.8.1 IP Spoofing Prevention.

Adversaries can leverage IP spoofing to impose a higher mitigation capacity demand on the first-wave victims. Thus, a critical component to defend against MTA is to prevent spoofed packets from congesting a first-wave victim's network link. Note that once a spoofed packet is leaked into the Internet, there is no silver bullet to identify its legitimacy [116]; transit ASes rely on ad hoc measures to mitigate the spoofed packets once they are leaked. For example, a network may establish private agreements with its neighboring networks to ensure that traffic with some source IPs must travel via certain links or tagged with some values to ensure its legitimacy [79]. While initiatives such as CAIDA's Spoofer project [22] and MANRS [121] are calling network operators to implement ingress/egress filtering to prevent IP spoofing, we need a much higher AS participation rate for an effective prevention.

5.8.2 Traffic Filter Capacity Expansion. The filter budget of a DDoS mitigation directly affects the number of second-wave victims. The existing DDoS mitigation research has two complementary ideas: machine-level filtering and Internet-level filtering. The machine-level filtering has two common approaches: 1) employs programmable switches to make an efficient use of low-latency memories (e.g., TCAM/CAM) [225, 110], and 2) uses commodity servers and efficient packet processing pipelines such as XDP [69]. The Internet-level is about building a filter distribution system so the participating ASes can each contribute some traffic filtering capacity. The cost to implement these two complementary ideas is not trivial and will take time. For example, machine-level filtering requires both training and development efforts of network operators other than its upfront equipment cost. Meanwhile, Internet-level filtering requires consensus from ASes to implement a common protocol.

5.9 Conclusion

Since the first documented attack over 20 years ago, the Internet continues to face severe large-scale DDoS attacks. While some ASes can over provision their computing and network resources to fight off large-scale attacks (e.g., subscribe to a DPS provider), most ASes rely on insufficient solutions to mitigate such attacks. The moving target attack, or MTA, exploits this mitigation insufficiency to wreak havoc on wide swaths of the Internet. Specifically, the attack leverages: (1) main ideas from prior attack research (i.e., pulsing and Crossfire) and (2) the practical attack and defense constraints to impose a mitigation dilemma on its first-wave victims, which may lead to a significant number of second-wave victims. Through experimentation on real-world data, we show that such an attack can simultaneously disconnect hundreds of ISP/NSP ASes or thousands of tier-3 ASes with a mere 100 Gbps attack. Furthermore, if these first-wave victims employ stateless fine-grained filtering to mitigate the attack, depending on the filter budget, they may end up disconnecting themselves from nearly 43.2% of the entire usable IPv4 space. Given the potential devastation MTA can unleash on today's Internet, we hope this chapter can raise the awareness of our network community and ultimately help spearhead the development and deployment of adequate DDoS mitigation solutions.

CHAPTER VI

DDOS SANDBOX – TOWARD HIGH-FIDELITY DDOS DEFENSE EVALUATION

Chapter IV and Chapter V exposed the practical issues with DDoS defense solutions from system input and system design perspectives. This chapter first highlights the glaring missing gaps in evaluating DDoS defense solutions. We then present an evaluation platform that addresses the missing gaps. Finally, we conduct experiments to demonstrate the correctness and efficiency of our platform and show that network operators and researchers can utilize the platform to evaluate DDoS attacks and defense solutions properly via a case study. Ultimately, we hope this platform can help its users choose and deploy matured defense solutions through empirical analysis to better defend against DDoS attacks.

The content in this chapter appeared in [178]. Lumin Shi is the leading author of this work and responsible for leading all the presented analyses.

6.1 Introduction

Advanced distributed denial-of-service (DDoS) attacks, such as the CrossFire attack [86] and CICADAS [88], seriously challenge the efficacy of the rudimentary DDoS defense strategies typically deployed by network operators. In particular, the most commonly deployed DDoS defense systems consist of two main components: a simple threshold-based DDoS detection/classification system, such as FastNetMon [49], and a coarsely grained mitigation solution such as Remotely Triggered Black Hole (RTBH) [97]. Unfortunately, these simple defense strategies struggle against the aforementioned, advanced DDoS attacks. A threshold-based detection solution rarely finds an appropriate balance between false positive and false negative rates, and a coarsely grained mitigation solution, by definition, filters legitimate traffic.

Despite over two decades of research to improve DDoS defense, network operators continue to choose these basic DDoS defense strategies regardless of their limitations. To draw insights into this discrepancy, Kokulu et al. recently surveyed security operation centers and found that network operators must obtain a thorough quantitative analysis of any security solution prior to deployment [95]. Without a comprehensive performance test of a potential security system in an environment similar to the network in which the system will deploy, the network operator cannot justify the risk associated with the adoption of new research defense systems. Clearly, to increase real-world DDoS defense deployment, the research community should conduct rigorous defense evaluation of DDoS defense solutions. However, by surveying well-received DDoS defense papers, we found that even highly-cited solutions frequently fail to evaluate their approach under realistic deployment scenarios. We highlight a few prominent missing gaps in evaluating DDoS defense solutions below.

Lack of insights into advanced attacks in action. While researchers continue to discover advanced DDoS attacks [86, 88, 175, 179], researchers rarely have the opportunity to study these attacks in action. To better defend against real-world attacks that leverage advanced attack mechanisms [30, 60], researchers should be able to run and study these attacks in a high-fidelity network environment. Preparing such a network environment is not a trivial task. For example, to correctly study pulsing attacks, such as CICADAS [88], that exploit TCP congestion control, the network must provide a traffic generator [171] to provide realistic background traffic that honors network condition changes (e.g., congestion control).

Lack of closed-loop networks for DDoS detection. As the most valuable outcome of a DDoS detection solution is to facilitate DDoS mitigation, we must evaluate the collaboration of different detection and mitigation solutions. In other words, rather than evaluate DDoS detection and mitigation separately, we need to keep all DDoS defense components in the loop to evaluate each component individually and the entire defense as a whole, thus a closed-loop evaluation. For example, a detection solution must work with abrupt network changes caused by the mitigation effort. Indeed, Vishwanath et al. [205] found many network systems, including DDoS detection solutions, can present biased conclusions if evaluated under non-closed-loop networks.

Lack of collateral damage analysis in filter-based mitigation. Despite the network community is slowly adopting fine-grained mitigation solutions for disseminating filtering rules (e.g., BGP Flowspec [123]), the limited hardware filter space [34, 83] is insufficient in mitigating DDoS attacks that do not contain common packet signatures [179, 85]. Network operators often use limited filters to mitigate attacks but at the cost of disabling access to non-attacking networks. Soldo et al. [189] proposed a set of filter generation methods to study the trade-off between limited filters and collateral damage. However, the performance of such these methods is highly dependent on the IP address locality. Therefore, we must provide an emulation environment with realistic IP addresses, only then can we evaluate such DDoS mitigation solutions with confidence.

To facilitate sound empirical evaluation of DDoS defense solutions, we propose, design, and develop an emulation platform for evaluating DDoS defense solutions with high fidelity. Referred to as the **DDoS SandBox** (or simply, the SandBox), it has the following capabilities:

1. Generation of inferred Internet topologies at the level of autonomous systems (ASes);
2. Assignment of realistic IP address spaces to ASes;
3. Routing and packet forwarding functions of each AS;
4. Packet-level mimicry of real network traffic that honors network condition changes (e.g.congestion); and
5. A simple usage model with high experiment portability.

The rest of this chapter is organized as follows. We first describe the related work in addressing the gaps (Sec. 6.2). We then present the design of our DDoS SandBox system (Sec. 6.3) and conduct a preliminary evaluation of a proof-of-concept implementation of the DDoS SandBox (Sec. 6.4). Finally, we conclude this chapter in Section 6.6.

6.2 Related Work: Emulation Systems

Dummynet [27] and Modelnet [203] are two early network emulation systems that support unmodified applications. They have fixed components that render the discussed missing gaps above difficult to close. For example, Modelnet employs Click [94], a software router, to configure and route traffic in the emulation network. This fixed design choice imposes burdens in deploying different routing implementations, which makes BGP-related DDoS defense techniques [1, 159, 135] difficult to evaluate. Later emulation platforms, such as described in [4, 102, 211, 157, 19], unanimously adopted Linux namespaces [119] to provide each process its own abstracted and isolated system resources with a single system kernel. For example, one type of Linux namespace, the Linux network namespace [118], can assign a unique network stack to each process, which allows the process to have its own routing table. Thus, network emulation tools can easily spawn "*hosts*" by assigning processes different network namespaces. Notably, mininet [102], a popular network emulation tool, also relies on Linux namespaces. Mininet then utilizes *cgroup* [117] to partition system resources to each process (e.g., set a maximum CPU utilization for a host – a process in its own namespaces). A more recent work, Containernet [157], extends mininet to support Docker. It encourages users to create self-containing software images to mitigate deployment issues.

A namespace-based emulation system offers low overhead when compared to virtual machines (VM) and many VM-based testbeds; the former emulates everything with a single operating system (OS) kernel while the latter emulates OS kernels but could incur additional interface translation and storage overheads. Our SandBox leverages Containernet to realize its capabilities. Specifically, the Docker support in Containernet allows us to create self-contained software images quickly, and its underlying mininet programming interfaces enable us to program the links between emulation nodes. These features create a solid foundation for us to close the missing gaps in a foreseeable amount of time.

6.3 DDoS SandBox

In this section, we first discuss the design considerations. We then introduce the main DDoS SandBox components at a high level. Finally, we detail the implementation of our proof of concept, which addresses the design considerations.

6.3.1 Design Considerations.

Using DDoS SandBox. The DDoS SandBox facilitates network operators to evaluate a defense solution in an emulation environment that mimics a real network. Meanwhile, DDoS researchers can also utilize the SandBox to gain insights into different DDoS attacks and defense solutions. Emulating a distributed system with a wide range of applications is a challenging task. Experimenters often run into issues such as fixing software dependencies or configuring network routers manually in an experiment. Thus, in the SandBox, we reduce the *deployment friction* by automating as many components as possible. In the meantime, the system remains fully customizable. For example, users can drop into a shell terminal of an arbitrary router node, and add or remove its network interfaces as they wish.

A mini Internet. Both advanced attacks and defense research projects require the emulation system to provide support for basic network functions (e.g., correct router ICMP behavior for *traceroute*). Ideally, the system should provide a *mini* Internet that is functionally equivalent to the real Internet. Only then can we study the latest DDoS attacks and evaluate defense solutions in a closed-loop environment. This includes but is not limited to 1) realistic AS-level IP space assignments based on the real-world BGP announcements, 2) BGP routing, and 3) closed-loop background traffic that reacts to the network conditions in real-time.

Elastic emulation fidelity. Today, anyone can have easy access to bare-metal servers with 100-core processors and 100s GBytes of memory from major cloud providers [182]. The modern hardware can easily emulate a small to medium network at high fidelity. However, emulation fidelity is not only about scalability. An experiment may require specific hardware for emulation (e.g., using a programmable switch for DDoS detection in the data plane). The emulation

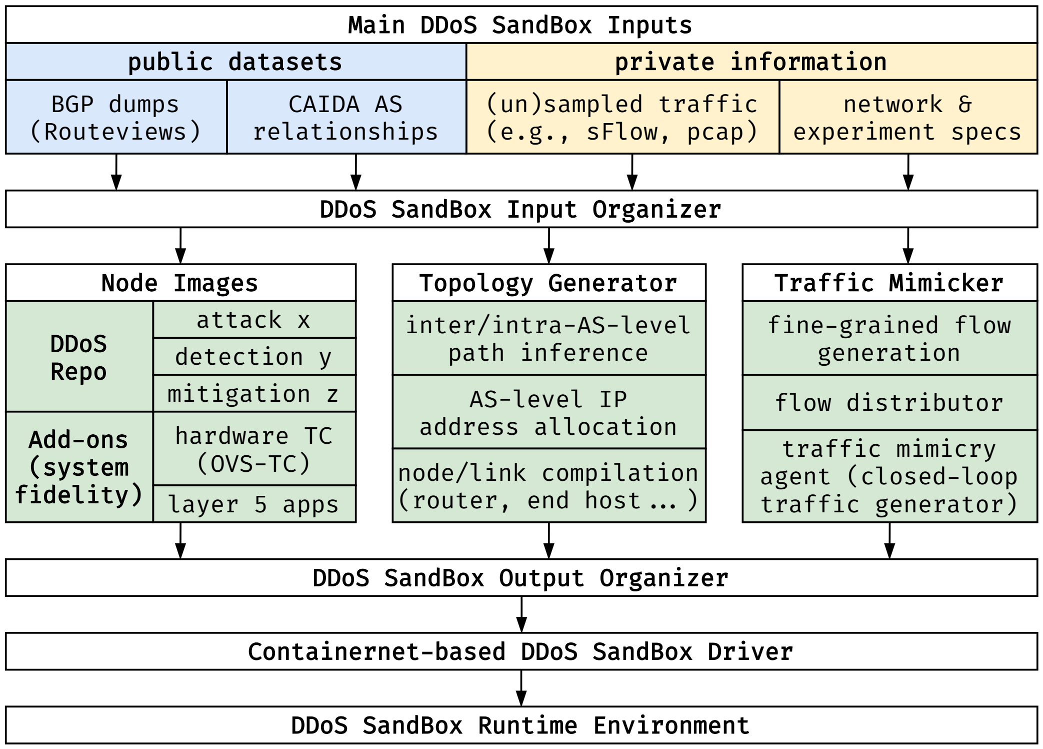

Main DDoS SandBox Inputs
public datasets
private information
BGP dumps (Routeviews)
CAIDA AS relationships
(un)sampled traffic (e.g., sFlow, pcap)
network & experiment specs
DDoS SandBox Input Organizer
Node Images
DDoS Repo
attack x
detection y
mitigation z
Add-ons (system fidelity)
hardware TC (OVS-TC)
layer 5 apps
Topology Generator
inter/intra-AS-level path inference
AS-level IP address allocation
node/link compilation (router, end host ...)
Traffic Mimicker
fine-grained flow generation
flow distributor
traffic mimicry agent (closed-loop traffic generator)
DDoS SandBox Output Organizer
Containernet-based DDoS SandBox Driver
DDoS SandBox Runtime Environment

Figure 39. DDoS SandBox Architecture

system must also consider supporting additional physical hardware, and as such, we can both offload the emulation load and increase the overall emulation fidelity.

6.3.2 System Components.

We introduce the system inputs and main SandBox components at a high level, as shown in Fig. 39. The blue colored boxes are open source software or dataset that anyone can acquire, the yellow colored boxes are information private to our users, and the green colored boxes are the main SandBox components.

Input. The SandBox creates a functional mini Internet with the following inputs: BGP table dumps from RouteViews [127], CAIDA AS relationships [25], and traffic traces (e.g., sFlow or a pcap trace, of a network). Typically, a user will input a benign traffic trace to generate the AS-level topology and background traffic in which the user can later inject attack traffic using the DDoS repository of attacks (which we describe further in the **Node Images** paragraph). However, the user can also leverage attack traces to study a specific instance of an attack. We reduce the input effort from users by automating data processing tasks for the public datasets. Additionally, users can feed network and experiment specifications to increase the emulation fidelity. For example, network operators can specify their intra-AS topologies and their upstream AS's partial intra-AS topologies to evaluate a defense solution of choice.

Topology Generator. The primary objective of Topology Generator is to create the blueprint of a mini Internet. Specifically, it leverages the user input data to: 1) create an inferred AS-level network that mimics a section of the Internet with regard to the input trace file, and assign realistic IP prefixes to each AS, 2) automate the network configuration for each network device (e.g., populate router and BGP configuration files that include information such as BGP prefix ownership and neighbors), and 3) attach nodes (e.g., traffic generators, DDoS bots, DDoS defense modules) to their belonging ASes. Of course, if the user provides detailed intra-AS topology information, the SandBox will reflect such information in the blueprint. Otherwise, we only create one router to represent an AS. While users are welcome to implement their own routers in the SandBox, we believe many users may not have strong preferences of the router implementation, as long as it is a realistic component. Thus, we provide a reference router that is based on Quagga routing suite [76]. Each Quagga router uses the following template to configure its network interfaces and announce BGP message to its neighboring routers.

```
router bgp {{ asn }}
    {% for network in networks %}
    network {{ network }}
    {% endfor %}
    {% for neighbor in neighbors %}
    neighbor {{ neighbor.ip }} remote-as {{ neighbor.asn }}
    {% endfor %}
log file {{ log_file | default("/tmp/bgpd.log", true) }}
debug bgp updates
```

Node Images. In the SandBox, we define any computational device, virtual or physical, as a node, and we package the software environment of the nodes as node images to improve the experiment portability. For example, if we instantiate a node using our reference router image, the node can 1) read/announce BGP prefix ownership populated by Topology Generator and 2) configure its forwarding table based on BGP announcements from its neighbors. Users can create a variety of node images containing any applications. For example, we can create an end host image with the ability to generate background network traffic by including a traffic generator. Similarly, to study different DDoS defense solutions, we plan to create a DDoS repository containing images of well-received attack strategies and defense solutions, thereby reducing the need for users to obtain attack traffic traces. We can then evaluate defense solutions under different attacks in realistic network environment.

Traffic Mimicker. Traffic Mimicker utilizes input traffic traces of a real-world network to create closed-loop background traffic in the SandBox. It generates fine-grained flow snapshots of ongoing flows based on the input traffic trace. For example, every second Traffic Mimicker may generate a snapshot of each 5-tuple network flow in the trace. Then, the flow distributor distributes a series of flow snapshots of each flow to the responsible end hosts with matched IPs. The example below shows a Comcast residential IP *23.24.100.123* communicating with *Amazon.com* at IP *205.251.242.103* (the snapshots were taken every second).

```
Example Output of Traffic Mimiker
['23.24.100.123', 2333, '205.251.242.103', 443,

 'TCP', 1, [50, 10, ...], [500, 1000, ...]]
```

We also see that the residential IP sends 50 KB of data to Amazon in the first second, then 10 KB of data in the next second. In return, Amazon.com sends 500 KB and then 1 MB of data back in two seconds. Finally, each traffic mimicry agent creates corresponding network sockets and communicates with the destination specified in each flow snapshot.

6.3.3 Proof of Concept (PoC) Implementation.

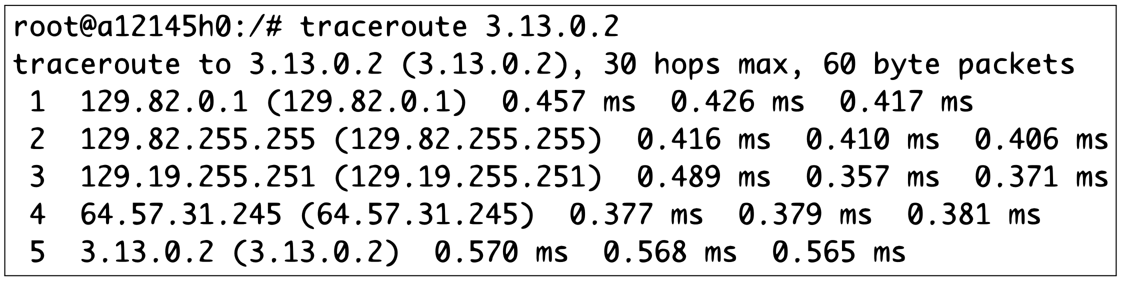
We choose Containernet as the PoC driver to implement the blueprint produced by Topology Generator. Specifically, we use Containernet to instantiate node images in Docker (e.g., the reference router image) and link instantiated nodes. The implementation also allows us to pass hardware network interfaces to a router node to achieve high-performance *traffic control* (*TC*) [120, 170] support. Such addition allows us to increase the emulation fidelity of DDoS SandBox.

6.4 Experiment: Creating a Mini Internet

6.4.1 Goals. To conduct a preliminary evaluation of our PoC, we first validate the correctness of Topology Generator via traceroute tests, and we then evaluate the single-host system scalability by analyzing network instantiation time with respect to the number of routers in the network. We plan on testing other aspects of the DDoS SandBox, such as the Traffic Mimicker, in our future work.

6.4.2 Setup. To create the AS topology, we feed a sampled sFlow trace from an IXP in the United States as our *only* private information input and two public information datasets (shown in Fig. 39) into the SandBox. Furthermore, we also attached several *traceroute*-enabled end hosts to two ASes within the AS topology. We conduct our evaluations on two distinct environments: 1) a Hyper-V VM utilizing three cores on an Intel i7-4970 processor with 24 GB of memory, and 2) a bare metal Amazon EC2 C5d instance utilizing 96 virtual cores with 192 GB of memory. Each Quagga router represents a single AS, and Docker version 19.03 is used to create and instantiate the router Docker image.

6.4.3 Traceroute Test. To test network connectivity and show that each AS is allocated realistic IP addresses, we perform several traceroute tests between ASes in the SandBox environment. An example traceroute is shown in Fig. 40. This traceroute (without *TC* policies) shows the route packets take between a randomly selected educational network and an arbitrary IP at a major cloud provider. The packets generated at the educational network flow through its upstream AS and Internet2 to reach the cloud IP. This reflects the real-world Internet AS topology, and we can find a corresponding AS-level path on *bgpview.io*. Furthermore, note that the SandBox provides a realistic IP assignment to each AS based on its real-world prefix ownership.



```
root@a12145h0:/# traceroute 3.13.0.2
traceroute to 3.13.0.2 (3.13.0.2), 30 hops max, 60 byte packets
 1  129.82.0.1 (129.82.0.1)  0.457 ms  0.426 ms  0.417 ms
 2  129.82.255.255 (129.82.255.255)  0.416 ms  0.410 ms  0.406 ms
 3  129.19.255.251 (129.19.255.251)  0.489 ms  0.357 ms  0.371 ms
 4  64.57.31.245 (64.57.31.245)  0.377 ms  0.379 ms  0.381 ms
 5  3.13.0.2 (3.13.0.2)  0.570 ms  0.568 ms  0.565 ms
```

Figure 40. An example traceroute test in DDoS SandBox

6.4.4 Network Instantiation Time. We benchmark the scalability of the DDoS SandBox by measuring the time it takes to: 1) create the reference router nodes, 2) populate configuration files for each node, 3) set up *veth* devices for each node, and 4) allocate IP addresses to each node. The main factor that affects network instantiation time is the number of nodes (or routers in this case) that need to be created in the SandBox. Note, there are many factors other than the number of nodes that can affect the system instantiation time, albeit to a smaller degree – for example, if an AS has a high degree of neighboring connections, the system may spend more time in creating *veth* devices and generating Quagga routing configuration files. Fig. 41 shows the network instantiation time with respect to the number of routers instantiated in the network. The two machines spend a similar amount of time to instantiate networks that have a limited amount of routers (i.e., less than 120). In fact, due to its high processor frequency, the 3-core Hyper-V VM performs slightly faster than the 96-core Amazon EC2 VM. However, for networks with more than 120 routers, the instantiation time for the Hyper-V machine increases exponentially due to its limited memory space, whereas the Amazon EC2 VM continues to follow a linear trend in instantiation time. Fig. 41 clearly shows that the SandBox can instantiate relatively large-scale networks within a relatively short amount of time.

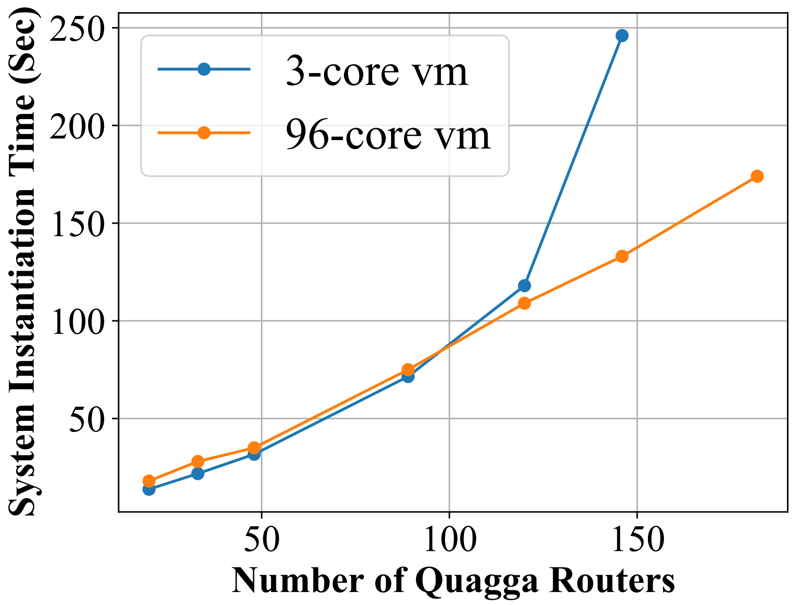


Figure 41. DDoS SandBox Instantiation Time

6.5 Experiment: A DDoS Attack and Defense Case Study

6.5.1 Goal. To demonstrate how users can employ DDoS SandBox to run DDoS emulations, we implement an attack and the defense against the attack with less than 200 lines of code. We go over the experiment setup along with a selected set of code snippets that implement the setup.

6.5.2 Setup. First, we create an AS-level partial Internet topology about UONET (AS3582). The following code snippet utilizes the `topology generator` of our system to create a blueprint with CAIDA prefix to ASN [23] and CAIDA AS relationships datasets. The system uses the valley-free model [114] to create the topology, and we set a limit on how many AS-level paths the system will implement.

```
UO_ASN = 3582
AS_PATH_LIMIT = 15  # Limit how many AS-level paths we implement
```

```
tg = TopoGenerator(
    target_asn = UO_ASN, topo_gen_mode = ASTopoGenMode.ONE_ROUTER,
    pfx2as_data_path = "routeviews-rv2-20220224-1000.pfx2as.gz",
    as_rel_data_path = "20220201.as-rel.txt.bz2",
    as_path_limit = AS_PATH_LIMIT
)
```

Within the UONET, we create a web server that returns a random amount of text for each HTTP request and five dummy hosts that bots will exploit for launching a carpet bombing attack [30].

```
# Create a web server at UO
as_uo = tg.as_dict[UO_ASN]
uo_webserver = as_uo.add_end_host(host_cls=Host_WebServer)
uo_webserver_ip =
 ↪  uo_webserver.net_interfaces[0].get_ip_interface().ip
# Create and save dummy hosts at UO
uo_dummy_hosts = []
for _ in range(5):
    uo_dummy_hosts.append(as_uo.add_end_host())
```

Next, we create six web clients and attach them to two remote ASes in the partial Internet topology. Each web client will fetch text from the web server periodically. At this point, we have a network environment that is ready to generate congestion-aware background traffic, albeit the traffic is not heterogeneous.

```
# Note that the first two AS paths of tg.as_paths are [UO] and [UO,
↪   UO's upstream].
# Therefore, we start from the third AS path (tg.as_paths[2]) to
↪   attach remote end hosts.
as_remote_1 = tg.as_dict[tg.as_paths[2][-1]]
as_remote_2 = tg.as_dict[tg.as_paths[3][-1]]
for _ in range(3):
    c1 = as_remote_1.add_end_host(host_cls=Host_WebClient)
    c2 = as_remote_2.add_end_host(host_cls=Host_WebClient)
```

We then create a set of bots to launch the carpet-bombing attack to congest the UONET's inbound link. Note that in a carpet-bombing attack, the traffic is dispersed through all IPs of the victim network. Therefore, the web server will not be able to detect much attack traffic.

```
# Create and save bots
bots = []
# Iterate through the ASes and attach a bot to each AS
for _ in range(2, AS_PATH_LIMIT):
    as_with_bot = tg.as_dict[tg.as_paths[i][-1]]
    """
    Pick random dummy hosts at UONET as attack targets.
    Each bot attacks one random dummy host.
    """
    dummy_host = random.choice(uo_dummy_hosts)
```

```
dummy_ip_to_attack =
 ↪  dummy_host.net_interfaces[0].get_ip_interface().ip
# Create a bot and limit each bot's maximum CPU quota
bot = as_with_bot.add_end_host(host_cls=Host_Bot,
    cpu_quota=25000, cpu_period=100000,
    environment={"dummy_attack_ip": f"{dummy_ip_to_attack}"}
)
bots.append(bot)
```

We record the received HTTP request rate and basic network statistics on the web server throughout the entire experiment. We also record network statistics on the UONET's upstream router to better capture the experiment. The web server is expected to experience drops in HTTP requests after the attack begins, and it will barely receive any traffic during the attack.

In this experiment, we assume UONET has a perfect DDoS detection accuracy, which identifies all IP addresses of the bots. We also assume UONET can request any remote ASes to drop the traffic from the bots. We implement the above assumptions with the following lines of code.

```
# Oracle: Gather the bot IPs we should block
denylist = [b.net_interfaces[0].get_ip_interface().ip for b in bots]
# Oracle: Gather the ASes that host the bots
attack_source_ases = set([b.owner_as for b in bots])
# Deploy denylist at each attack source AS.
for attack_source_as in attack_source_ases:
```

```
# Get the router at each AS that hosts bots
router = attack_source_as.routers[0]
router_container = adapter.sb_router_to_container[router]
# Deploy the denylist
for bot_ip in denylist:
    mitigation_cmd = f"nohup ip rule add blackhole from {bot_ip}
     ↪  &"
    router_container.cmd(mitigation_cmd)
```

6.5.3 Experiment Results. We now report the collected data from the web server and UONET's upstream router in the emulation. The experiment lasts for four minutes, and it includes four stages: (1) Network with only the web traffic, (2) Network with both the web and attack traffic, (3) Network with mitigation in place, and (4) Bots stop their attack. Each stage lasts for exactly one minute. We perform the fourth stage to understand if we spent too much resource on the attack traffic generation that affects the emulation fidelity. Specifically, if the measurement results show drastically different behavior in the third stage and the fourth stage, the system has a low emulation fidelity.

During the first stage, the web server observes approximately five to six HTTP requests per second, as shown in Fig. 42. The request rate drops sharply as soon as the bots start to attack UONET. Because the bots only send attack traffic towards the dummy hosts at UONET, the web server does not receive any attack traffic but benign traffic from its clients.

Indeed, Fig. 43 shows that effective network throughput of the web server greatly reduces from the 60-second mark. Meanwhile, the effective HTTP request

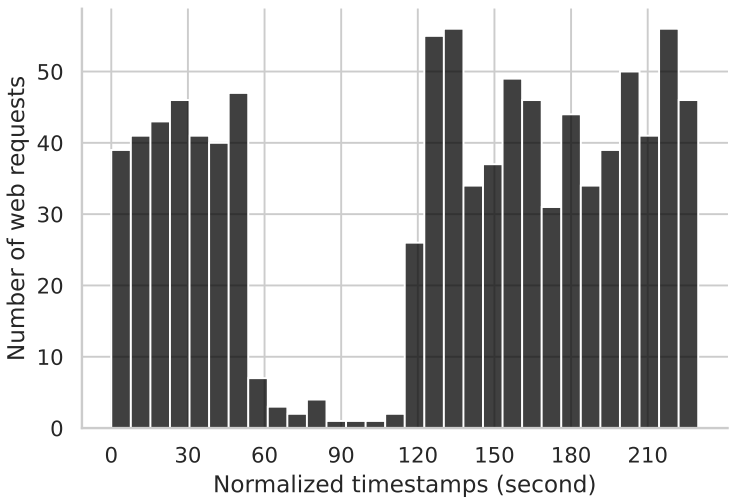


Figure 42. The HTTP request rate observed by the web server at UONET.

rate is reduced to 2.5% of its normal rate. These numbers suggest that the bots launched a successful link-flooding attack against UONET. From the 120-second mark, we requested all remote ASes that host bots to drop attack traffic. We see the web server has a gradual increase in network throughput after the mitigation is in place. We do not see an instant jump in network throughput — it takes a moment for the benign web clients need to break out of their TCP exponential backoff stage. At the 180-second mark, we inform all bots to stop the attack. We see the network throughput of the web server stay generally the same in the third and fourth experiment stages. In other words, the emulation fidelity holds.

We now examine the network statistics of UONET's upstream router, as shown in Figure 44. In the first minute, the upstream router receives and transmits mostly the traffic from and to the web server, as well as the BGP announcements created by routers in the emulation environment. From the 60-second mark to the 120-second mark, the router sees a drastic increase in the amount of traffic it receives. Specifically, the router receives traffic at 3.7 Gbps, as shown in Fig. 44a.

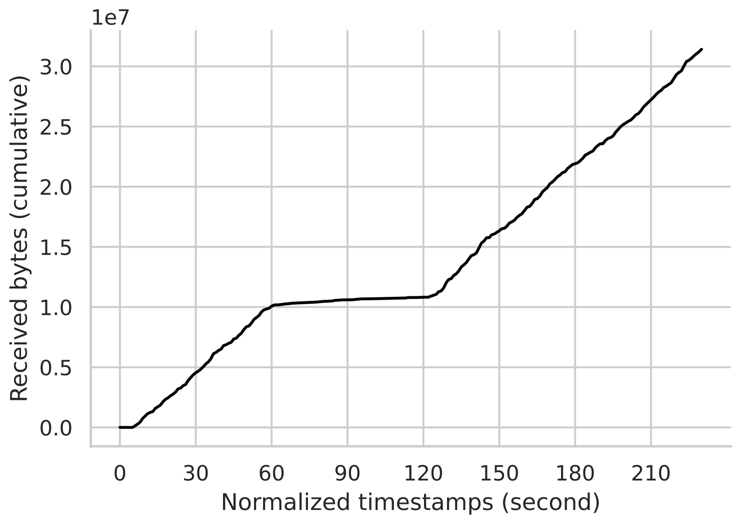
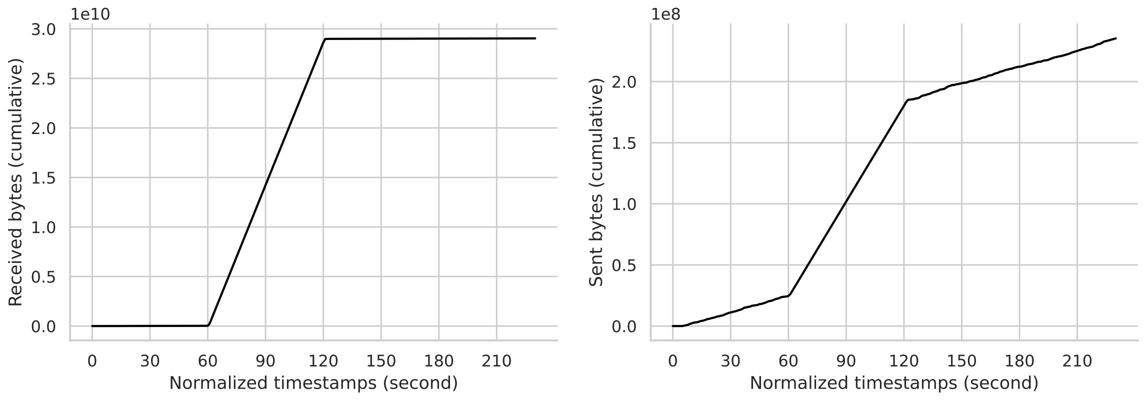


Figure 43. The total received bytes over time observed by the web server at UONET.

Note that because the base unit of the Y axis is in 10s of gigabytes, the web traffic is negligible. Therefore, it appears the upstream router does not receive any traffic in the first and the last two minutes. However, the link bandwidth between UONET and the upstream router is limited at 20 Mbps in the experiment, as shown in Fig. 44b.



(a) The total received bytes over time. (b) The total transmitted bytes over time.

Figure 44. Network statistics of UONET's upstream router.

6.6 Conclusion

The DDoS SandBox is an ongoing project that is under development. We are integrating more reference node images into the SandBox (e.g., Traffic Mimicker) We are also investigating in reducing the required efforts to include physical devices. We plan to experiment with the scalability of the SandBox across multiple servers and investigate different approaches to extrapolate experiment results to the Internet scale. In fact, many container network interface (CNI) plugins [37] already provides us a transparent layer that allows us to run containers across multiple machines. Our ultimate goal for the DDoS SandBox is to run sound empirical DDoS attack and defense experiments. We hope results derived from the SandBox can draw more attention from network operators, and pave the way for the deployment of various defense systems. We will first implement the advanced DDoS attacks and well-received defense solutions, and provide them as reference node images in the SandBox for users to run general DDoS experiments. To the best of our knowledge, this is the first attempt to bridge the missing gaps for sound empirical DDoS experiments, and we have shown initial success in bridging those gaps in our PoC. Our PoC system is open source, and for more design details, the code is available on: `https://github.com/DDoS-SandBox`.

CHAPTER VII

FUTURE WORK

In the following we list several possible directions to further the research presented in this dissertation.

7.1 On Capturing DDoS Traffic Footprints on the Internet

While the chapter considers many real-world operational constraints in a network, we can still improve the PathFinder system in a pilot study that involves multiple ASes of different sizes. Specifically, we can gather empirical results to understand/estimate the required system capacity for small, medium, and large transit networks to deploy the system. Furthermore, we can extend the system with an IP spoofing inference module to help DDoS defense systems to make better mitigation decisions with a constrained budget.

7.2 On Quantifying Multi-Wave Victims in DDoS Mitigation

We can further extend this chapter from several perspectives as follows. First, there is room for optimizing the moving target attack cycle so that MTA can disconnect more first-wave victims. For example, an adversary can develop attack strategies that consider the timezone of the first-wave victims; depending on the time of the day, ASes may prefer fewer network jitters over total available link bandwidth in attack mitigation.

Second, we can further improve the traffic generation module to save the attack cost, should the adversary needs to prioritize second-wave victims. For example, suppose the adversary wants to introduce more cloud providers as second-wave victims in an attack. In that case, the adversary needs to rent hosts from the cloud providers to launch MTA effectively. However, cloud providers often meter their hosts' outbound traffic and impose additional costs when the hosts send traffic over the limit. Therefore, the adversary may leverage bots with IP spoofing capability to send attack traffic as if the traffic is from the cloud providers, yet still, bypass stateful defenses by asking the cloud provider hosts to share the state information (e.g., TCP sequence number).

Last but not least, The existing papers analyze multi-wave victims at IP and AS levels. However, not all IPs or ASes are equal; some IPs or ASes are responsible for serving more users than others. For example, in densely populated countries that lack IPv4 addresses, each IP connects more users to the Internet than in loosely populated countries. Therefore, we need to consider other more representative metrics for evaluating the victims.

7.3 DDoS SandBox

The SandBox project is a preliminary step towards the sound, empirical evaluation for DDoS defense projects. We showcase a container-based system that addresses the glaring missing gaps in existing DDoS defense evaluation. Nevertheless, we can further the research in many areas as follows. First, we may extend the system to distribute containers over many machines to support large-scale DDoS experiments. Such an extension allows experimenters to launch more end-host application processes (e.g., web browsers) to generate even more realistic, congestion-aware background traffic without affecting the overall network throughput of the system. Second, the reference images for DDoS attacks and defense solutions. The main goal of the SandBox project is to facilitate its users to benchmark various defense solutions in a controlled yet realistic environment. We hope to implement well-received DDoS attacks and defense solutions as a set of reference images for the users to experiment with and recipes for creating reproducible DDoS experiments.

CHAPTER VIII

CONCLUSIONS

The Internet remains challenged by DDoS attacks as they evolve after more than twenty years. The scale of attacks is not only larger than ever, but attacks are harder to detect and mitigate. Nevertheless, the Internet's fundamental design, machines are free to send traffic to any other machines, remains the same. There are numerous reasons for the above challenge. First, given the sheer amount (70,000+) of the ASes worldwide, it is impractical for every AS to accept a general defense solution to defend against large-scale attacks. Second, ASes lack incentives to be committed to addressing large-scale DDoS attacks. Indeed, because some ASes have not experienced large-scale attacks, they may not take the problem seriously or know the latest defense solutions. Last but not least, it is likely that the existing DDoS defense solutions are not mature enough and require substantial maintenance efforts that are unacceptable by many ASes.

In this thesis we investigated the existing DDoS defense solutions to understand the different defense directions and issues that remain relevant today. We then proposed solutions to improve the system *input*, *design*, and *evaluation* of DDoS defense. The thesis shows why DDoS defense systems need a better view of the Internet's traffic at AS level to fight off attack efficiently. It then presents a novel attack to expose the inefficiencies in the existing defense systems; we use the attack as a lens to quantify the damage it inflicts to a large swathe of the Internet. Finally, it argues why a defense solution needs a sound empirical evaluation and proposes a platform to facilitate DDoS evaluation that mimics real-world networks.

More specifically, we have made the following contributions in each chapter.

In Chapter IV, we proposed the PathFinder system as a service to facilitate effective DDoS defense. A DDoS defense system can request the PathFinder system on behalf of a DDoS victim the victim's traffic footprints. We carefully design the PathFinder architecture to be easily deployable on today's Internet: ASes can join the PathFinder system without relying on other ASes. Each AS operates an on-demand service model which introduces a low overhead, and PathFinder does not require every AS to participate in helping DDoS defense. Next, we make the design of PathFinder with a series of real-world operational factors in mind. We design a new data structure called PFTrie that records traffic footprints at a line rate with optimization techniques. We also design a streaming mechanism that manages the delivery of traffic footprints, and it also lowers the transmission overhead. PathFinder can dynamically adjust its traffic footprints granularity to reduce the system load when necessary. Finally, we evaluate PathFinder and show that PathFinder can greatly improve the performance of DDoS defense. Its PFTrie records traffic footprints in real-time at a manageable overhead. Its streaming and zooming mechanism incur low transmission latency.

In Chapter V, we present the moving target attack (MTA), a link-flooding attack that introduces a mitigation dilemma to many first-wave victims. Each victim either endures the attack or disconnect itself from a wide spread of second-wave victims during DDoS defense. The second-wave victims include networks that contain benign hosts. MTA is a practical and novel attack that exposes real-world DDoS mitigation issues quantitatively; this is the first work that studies the consequences of launching a DDoS attack against multiple ASes simultaneously. We surveyed network operators to learn their DDoS mitigation solutions and the limitations of the solutions and validated our assumptions in this chapter.

We evaluate MTA against real-world mitigation solutions in simulation and demonstrate that the attack can inflict severe damage to today's Internet. In many scenarios, we show that first-wave victims have to disconnect themselves from nearly half of the usable IPv4 address space to mitigate the attack. The chapter then summarizes two critical countermeasures for building a more resilient Internet against DDoS attacks.

Chapter VI showcases the DDoS SandBox, an emulation platform that enables its users (e.g., network operators, researchers) to conduct high-fidelity DDoS research. To identify the impact of a defense solution in realistic network settings, our platform offers to emulate a mini Internet topology with realistic IP address space allocation and generate representational, closed-loop background traffic specific to particular networks. As such, our platform fulfills the prominent missing gaps in current DDoS research. Finally, we conduct experiments to demonstrate the correctness and scalability of our platform, as well as a case study that provides a basic bootstrapping script for evaluating a specific link-flooding attack and the defense against the attack.

DDoS defense is a research area that requires more than a deep understanding of computer networks to advance its state; non-technical factors are precious to further the research. We hope this thesis work contributes some new perspectives in the area.

APPENDIX
THE MOVING TARGET ATTACK

A.1 DDoS Mitigation Survey Results

Please visit `https://docs.google.com/spreadsheets/d/e/2PACX-1vS5YAD6ZzXDtcGNaOPBXftCkyuhgCSYdXzkjAX9Xo9Nbwqsh4ibz7_EXk-khSmFWnplj6t5fFBwjMbe/pubhtml` to browse our survey questions and results.

A.2 Major DPS Provider ASNs

We list the DPS providers and their ASNs that announce BGP messages as follows:

- Akamai (Prolexic): 32787
- Cloudflare: 13335
- DDoS-Guard: 57724
- Defense.net: 55002
- Incapsula: 19551
- Neustar: 19905
- Nexusguard: 45474

A.3 IP Spoofing and Stateful Firewall

In IPv4 network, we can spoof $N_{sip} = 2^{32}$ source IP addresses. MTA can disperse its traffic among all IP addresses of a first-wave victim (denoted as N_{dip}). Other than IP address fields, a stateful solution also tracks the source and destination ports, each contains up to 2^{16} possible ports (denoted as N_{sp}, and N_{dp}, respectively). In theory, the four variables above allows MTA to generate

$N_{sip} * N_{dip} * N_{sp} * N_{dp}$ unique TCP packets. The stateful solution must forward each unique SYN packet in order to track its future states. For example, if the first-wave victim own a /16 network prefix, the stateful mitigation needs to handle $2^{32+16+16+16} = 2^{80}$ unique states; no device is powerful enough to handle so many states.

Suppose we want to attack a stub network for 24 hours with 100Gbps of bandwidth. If all bots send traffic at smallest TCP packet size (i.e., 84 bytes), we need

$$\begin{aligned} 100\ Gbps * 10^9 \frac{bps}{Gbps} / 8 \frac{bits}{byte} & * 24\ hrs * 3600 \frac{secs}{hr} / 84 \frac{bytes}{packet} \\ & \approx 1.286 * 10^{13} \end{aligned}$$

packets. Despite the huge number of unique packets such an attack requires, it would take $2^{80}/(1.286 * 10^{13}) \approx 94$ billion days to cycle through all unique packets. While the above calculation does not exclude bogon IP blocks nor the network policies a first-wave victim may have in place, it shows the limitation of a stateful mitigation solution when facing IP spoofing nonetheless.

A.4 Inferred Traffic Levels

The PeeringDB dataset contains 16 traffic levels that range anywhere from 20 Mbps to 100 Tbps. However, the number of available data points (i.e., the 7,239 tier-3 ASes with traffic information) is not enough to provide accurate prediction over the 16 categories. Figure 30 shows that the distribution of the data points are highly skewed. Specifically, 26.2% of the data points have at most 1Gbps of traffic level, 61.5% of them have at most 5Gbps of traffic level, 78.6% of them have at most 10Gbps of traffic level, 89.5% of them have at most 20Gbps of traffic level, and 95.1% of them have at most 50Gbps of traffic level. Worse, the average *cosine* *similarity* score of the tier-3 ASes's features with under 10Gbps is over 93.9%, which means that the classification algorithms cannot accurately infer the traffic levels under 10 Gbps. Therefore, we reduced the number of traffic levels from 16 to 4, and they are: ≤ 10 *Gbps*, ≤ 20 *Gbps*, ≤ 50 *Gbps*, and ≥ 50 *Gbps*.

A.5 Filter Generation Process

As input, the process takes the maximum number of filters that can be generated (i.e., filter budget), the minimum amount of traffic that needs to be filtered (i.e., required coverage), a set of attack IPs (/32), and the weight of each IP (e.g., the amount of traffic generated by an IP). The algorithm first generates a set of filters of size /32 to cover each attack IP. Then the filter selection process begins. First, the algorithm chooses a subset of filters such that the filter budget is not surpassed and the amount of attack traffic filtered is maximized. If this subset of filters meets the required coverage, the algorithm returns the subset and terminates. If this subset does not meet the required coverage, the algorithm begins the aggregation process, where it attempts to aggregate filter in the complete set of filters. In each round of aggregation, the algorithm attempts to decrease a filter's IP prefix length to the next smallest prefix length (first round: /31, second round: /30, etc.) thereby generating a new filter (e.g., if a filter has a prefix of /32, then in the first round, the algorithm attempts to generate a new filter with a prefix of /31 that includes the current filter of prefix /32). A new filter is generated only if it filters more attack traffic than any other possible new filter that could be generated. After a new filter is generated, the algorithm conducts the filter selection process to check if the required coverage is now met. If not, the algorithm continues to generate new filter. If no new filter can be generated of the given prefix size to meet the required coverage, the algorithm begins a new round of aggregation, this time with the next smallest prefix length. This entire process continues until the required coverage is met.

BIBLIOGRAPHY

[1] Abley, J., and Lindqvist, K. E. Operation of Anycast Services. `https://tools.ietf.org/html/rfc4786`, 2006.

[2] Afek, Y., Bremler-Barr, A., and Feibish, S. L. Zero-Day Signature Extraction for High-Volume Attacks. *IEEE/ACM Transactions on Networking 27*, 2 (April 2019).

[3] Aghaei-Foroushani, V., and Zincir-Heywood, A. N. Autonomous System Based Flow Marking Scheme for IP-Traceback. In *IEEE/IFIP Network Operations and Management Symposium* (2016).

[4] Ahrenholz, J. Comparison of core network emulation platforms. In *Military Communications Conference (MILCOM)* (2010).

[5] Akamai. 2020 State of the Internet Security: Financial Services – Hostile Takeover Attempts. `https://www.akamai.com/us/en/resources/our-thinking/state-of-the-internet-report/global-state-of-the-internet-security-ddos-attack-reports.jsp`, 2020.

[6] Akamai. Prolexic Routed. `https://www.akamai.com/us/en/products/security/prolexic-solutions.jsp`, 2021.

[7] Andersen, D. G. Mayday: Distributed Filtering for Internet Services. In *Proceedings of the 4th Conference on USENIX Symposium on Internet Technologies and Systems* (2003), vol. 4 of *USITS'03*.

[8] Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., et al. Understanding the Mirai Botnet. In *26th USENIX Security Symposium (USENIX Security 17)* (2017).

[9] Argyraki, K., and Cheriton, D. R. Active Internet Traffic Filtering: Real-time Response to Denial of Service Attacks. In *USENIX Annual Technical Conference* (2005), vol. 38.

[10] Argyraki, K., and Cheriton, D. R. Scalable Network-Layer Defense Against Internet Bandwidth-Flooding Attacks. *IEEE/ACM Transactions on Networking 17*, 4 (2009).

[11] Argyraki, K. J., and Cheriton, D. R. Active Internet Traffic Filtering: Real-Time Response to Denial-of-Service Attacks. In *USENIX Annual Technical Conference* (2005).

[12] Armbruster, B., Smith, J. C., and Park, K. A packet filter placement problem with application to defense against spoofed denial of service attacks. *European Journal of Operational Research 176*, 2 (2007).

[13] Ballani, H., and Chawathe, Y. Off by Default! *ACM Workshop on Hot Topics in Networks* (2005).

[14] Basescu, C., Reischuk, R. M., Szalachowski, P., Perrig, A., Zhang, Y., Hsiao, H.-C., Kubota, A., and Urakawa, J. SIBRA: Scalable Internet Bandwidth Reservation Architecture. In *Network and Distributed System Security Symposium* (2016), NDSS'16.

[15] Behal, S., and Kumar, K. Detection of DDoS Attacks and Flash Events Using Novel Information Theory Metrics. *Computer Networks 116* (2017).

[16] BGPKIT. BGP Data Analysis Tool Kit. `https://github.com/bgpkit`, 2021.

[17] Blog, G. S. A javascript-based ddos attack as seen by safe browsing. `https://security.googleblog.com/2015/04/a-javascript-based-ddos-attack-as-seen.html`, 2015.

[18] Bloom, B. H. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM 13*, 7 (jul 1970).

[19] Bonofiglio, G., Iovinella, V., Lospoto, G., and Di Battista, G. Kathará: A container-based framework for implementing network function virtualization and software defined networks. In *2018 IEEE/IFIP Network Operations and Management Symposium* (2018).

[20] Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., and Varoquaux, G. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning* (2013).

[21] Burch, H., and Cheswick, B. Tracing Anonymous Packets to Their Approximate Source. In *USENIX Large Installation System Administration Conference* (2000).

[22] CAIDA. State of IP Spoofing. `https://spoofer.caida.org/summary.php`, 2019.

[23] CAIDA. Routeviews Prefix to AS mappings Dataset for IPv4 and IPv6. `https://www.caida.org/catalog/datasets/routeviews-prefix2as/`, 2021.

[24] CAIDA. The CAIDA AS Organizations Dataset. `http://www.caida.org/data/as-organizations`, 2021.

[25] CAIDA. The CAIDA AS Relationships Dataset. `http://www.caida.org/data/active/as-relationships`, 2021.

[26] Campbell, B., Bradley, J., Sakimura, N., and Lodderstedt, T. OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens. `https://rfc-editor.org/rfc/rfc8705.txt`, 2020.

[27] Carbone, M., and Rizzo, L. Dummynet revisited. *ACM SIGCOMM Computer Communication Review 40*, 2 (2010).

[28] Catalin Cimpanu. You Can Now Rent a Mirai Botnet of 400,000 Bots. `https://www.bleepingcomputer.com/news/security/you-can-now-rent-a-mirai-botnet-of-400-000-bots/`, 2018.

[29] Census Bureau. Census Bureau QuickFacts. `https://web.archive.org/web/20220107213155/https://www.census.gov/quickfacts/fact/table/losangelescitycalifornia,US/VET605219`, 2022.

[30] Cimpanu, C. 'Carpet-bombing' DDoS attack takes down South African ISP for an entire day. `https://www.zdnet.com/article/carpet-bombing-ddos-attack-takes-down-south-african-isp-for-an-entire-day/`, 2019.

[31] Cimpanu, C. AWS said it mitigated a 2.3 Tbps DDoS attack, the largest ever. `https://www.zdnet.com/article/aws-said-it-mitigated-a-2-3-tbps-ddos-attack-the-largest-ever`, 2020.

[32] Cisco. Configuring Secure Shell on Routers and Switches Running Cisco IOS. `http://www.cisco.com/c/en/us/support/docs/security-vpn/secure-shell-ssh/4145-ssh.html`, 2007.

[33] Cisco. Cisco IOS IP Routing: BGP Command Reference. `http://www.cisco.com/c/en/us/td/docs/ios/iproute_bgp/command/reference/irg_book/irg_bgp5.html`, 2013.

[34] Cisco. IP Addresses and Services Configuration Guide for Cisco NCS 540 Series Routers, IOS XR Release 6.3.x. `https://www.cisco.com/c/en/us/td/docs/iosxr/ncs5xx/ipaddress/63x/b-ip-addresses-cg-63x-ncs5xx/b-ip-addresses-cg-63x-ncs5xx_chapter_01.html`, 2018.

[35] Cisco. Catalyst Switched Port Analyzer (SPAN) Configuration Example. `http://www.cisco.com/c/en/us/support/docs/switches/catalyst-6500-series-switches/10570-41.html`, 2019.

[36] Cisco. Cisco Nexus 9000 Series NX-OS Interfaces Configuration Guide, Release 7.x . https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus9000/sw/7-x/interfaces/configuration/guide/b_Cisco_Nexus_9000_Series_NX-OS_Interfaces_Configuration_Guide_7x/b_Cisco_Nexus_9000_Series_NX-OS_Interfaces_Configuration_Guide_7x_chapter_0111.html, 2020.

[37] Cloud Native Computing Foundation. CNI - the Container Network Interface. https://github.com/containernetworking/cni, 2020.

[38] Cloudflare. What is a DNS Flood? https://www.cloudflare.com/learning/ddos/dns-flood-ddos-attack/, 2019.

[39] Cloudflare. What is an HTTP flood DDoS attack? https://www.cloudflare.com/learning/ddos/http-flood-ddos-attack/, 2019.

[40] Cloudflare. Cloudflare Magic Transit. https://www.cloudflare.com/magic-transit, 2021.

[41] Corin, R. D., Costanzo, A., Callegati, F., and Siracusa, D. Methods and techniques for dynamic deployability of software-defined security services. *CoRR* (2020).

[42] Debroy, S., Calyam, P., Nguyen, M., Stage, A., and Georgiev, V. Frequency-minimal moving target defense using software-defined networking. In *2016 International Conference on Computing, Networking and Communications (ICNC)* (Feb 2016).

[43] Dhawan, M., Poddar, R., Mahajan, K., and Mann, V. SPHINX: Detecting Security Attacks in Software-Defined Networks. In *Network and Distributed System Security Symposium* (2015).

[44] Dietzel, C., Wichtlhuber, M., Smaragdakis, G., and Feldmann, A. Stellar: Network Attack Mitigation Using Advanced Blackholing. In *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies* (2018), CoNEXT'18.

[45] Division, C. 1996 cert advisories. https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=496170, 1996.

[46] Dixon, C., Anderson, T., and Krishnamurthy, A. Phalanx: Withstanding Multimillion-Node Botnets. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation* (2008), NSDI'08.

[47] Eddy, W. M. Defenses Against TCP SYN Flooding Attacks. https://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-34/syn-flooding-attacks.html, 2006.

[48] Emerging Technology from the arXiv. The first DDoS attack was 20 years ago. This is what we've learned since. https://www.technologyreview.com/2019/04/18/103186/the-first-ddos-attack-was-20-years-ago-this-is-what-weve-learned-since, 2019.

[49] FastNetMon. Testimonials. https://fastnetmon.com/client-testimonials, 2020.

[50] Fayaz, S. K., Tobioka, Y., Sekar, V., and Bailey, M. Bohatei: Flexible and Elastic DDoS Defense. In *24th USENIX Security Symposium (USENIX Security 15)* (2015).

[51] Feng, Y., Li, J., and Nguyen, T. Application-layer ddos defense with reinforcement learning. In *Proceedings of the International Symposium on Quality of Service (IWQoS)* (2020).

[52] Ferguson, P., and Senie, D. Network Ingress Filtering: Defeating Denial-of-Service Attacks Which Employ IP Source Address Spoofing. Tech. rep., 2000.

[53] Flavel, A., Mani, P., Maltz, D., Holt, N., Liu, J., Chen, Y., and Surmachev, O. FastRoute: A Scalable Load-Aware Anycast Routing Architecture for Modern CDNs. In *12th USENIX Symposium on Networked Systems Design and Implementation* (2015), NSDI'15.

[54] Ford, A., Raiciu, C., Handley, M., Bonaventure, O., and Paasch, C. TCP extensions for multipath operation with multiple addresses. Tech. rep., 2019.

[55] François, J., Aib, I., and Boutaba, R. FireCol: A Collaborative Protection Network for the Detection of Flooding DDoS Attacks. *IEEE/ACM Transactions on Networking 20*, 6 (Dec 2012).

[56] Freedman, M. J., Lakshminarayanan, K., and Mazières, D. OASIS: Anycast for Any Service. In *Networked Systems Design and Implementation* (2006), NSDI'06.

[57] FRRouting. FRRouting User Guide. http://docs.frrouting.org/en/latest/bgp.html#displaying-bgp-information, 2020.

[58] Ganti, V., and Yoachimik, O. Network-layer DDoS attack trends for Q4 2020 . `https://blog.cloudflare.com/network-layer-ddos-attack-trends-for-q4-2020/`, 2021.

[59] Gartner. DDoS Mitigation Services Reviews and Ratings. `https://www.gartner.com/reviews/market/ddos-mitigation-services`, 2021.

[60] Gavrichenkov, T. Reflection DDoS last week (was: syn flood attacks from NL-based netblocks). `https://seclists.org/nanog/2019/Aug/415`, 2019.

[61] Gavrilis, D., and Dermatas, E. Real-Time Detection of Distributed Denial-of-Service Attacks Using RBF Networks and Statistical Features. *Computer Networks 48*, 2 (2005).

[62] Gilad, Y., Herzberg, A., Sudkovitch, M., and Goberman, M. CDN-on-Demand: An Affordable DDoS Defense via Untrusted Clouds. *Network and Distributed System Security Symposium (NDSS'16)* (2016).

[63] Graham, R. MASSCAN: Mass IP port scanner. `https://github.com/robertdavidgraham/masscan`, 2021.

[64] Guirguis, M., Bestavros, A., and Matta, I. Exploiting the transients of adaptation for roq attacks on internet resources. In *Proceedings of the 12th IEEE International Conference on Network Protocols (ICNP)* (2004).

[65] Gummadi, R., Balakrishnan, H., Maniatis, P., and Ratnasamy, S. Not-a-bot: Improving service availability in the face of botnet attacks. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI'09)* (2009).

[66] Gupta, A., Vanbever, L., Shahbaz, M., Donovan, S. P., Schlinker, B., Feamster, N., Rexford, J., Shenker, S., Clark, R., and Katz-Bassett, E. SDX: A Software Defined Internet Exchange. vol. 44.

[67] Hassidim, A., Raz, D., Segalov, M., and Shaqed, A. Network utilization: The flow view. In *IEEE INFOCOM* (2013).

[68] Hilgenstieler, E., Duarte, E. P., Mansfield-Keeni, G., and Shiratori, N. Extensions to the Source Path Isolation Engine for Precise and Efficient Log-Based IP Traceback. *Computers & Security 29* (2010).

[69] Høiland-Jørgensen, T., Brouer, J. D., Borkmann, D., Fastabend, J., Herbert, T., Ahern, D., and Miller, D. The express data path: Fast programmable packet processing in the operating system kernel. In *Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies* (2018).

[70] Hsiao, H.-C., Kim, T. H.-J., Yoo, S., Zhang, X., Lee, S. B., Gligor, V., and Perrig, A. STRIDE: Sanctuary trail - refuge from internet ddos entrapment. In *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security* (2013), ASIA CCS '13.

[71] Huici, F., and Handley, M. An Edge-to-Edge Filtering Architecture Against DoS. *SIGCOMM Computer Communication Review 37*, 2 (2007).

[72] Hussain, A., Heidemann, J., and Papadopoulos, C. A Framework for Classifying Denial of Service Attacks. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (2003), SIGCOMM '03.

[73] Hussain, A., Heidemann, J., and Papadopoulos, C. Identification of Repeated Denial of Service Attacks. In *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications* (2006).

[74] Imperva. R.U.D.Y. (R-U-Dead-Yet?). `https://www.imperva.com/learn/application-security/rudy-r-u-dead-yet/`, 2019.

[75] Imperva. DDoS Protection for Networks. `https://www.imperva.com/products/infrastructure-ddos-protection-services/`, 2021.

[76] Jakma, P., and Lamparter, D. Introduction to the quagga routing suite. *IEEE Network 28*, 2 (2014), 42–48.

[77] Jia, Q., Sun, K., and Stavrou, A. MOTAG: Moving Target Defense against Internet Denial of Service Attacks. In *2013 22nd International Conference on Computer Communication and Networks (ICCCN)* (July 2013).

[78] Jia, Q., Wang, H., Fleck, D., Li, F., Stavrou, A., and Powell, W. Catch Me if You Can: A Cloud-Enabled DDoS Defense. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks* (2014), pp. 264–275.

[79] Jin, C., Wang, H., and Shin, K. G. Hop-Count Filtering: An Effective Defense against Spoofed DDoS Traffic. In *Proceedings of the 10th ACM Conference on Computer and Communications Security* (2003), CCS '03.

[80] Jin, L., Hao, S., Wang, H., and Cotton, C. Your Remnant Tells Secret: Residual Resolution in DDoS Protection Services. In *Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)* (2018).

[81] Juniper. Example: Configuring Port Mirroring for Local Monitoring of Employee Resource Use on EX Series Switches. `https://www.juniper.net/documentation/en_US/junos/topics/example/port-mirroring-local-ex-series.html`, 2018.

[82] Juniper. Configuring SSH Service for Remote Access to the Router or Switch. `https://www.juniper.net/documentation/en_US/junos/topics/task/configuration/ssh-services-configuring.html`, 2019.

[83] Juniper. Firewall Filters for EX Series Switches Overview. `https://www.juniper.net/documentation/en_US/junos/topics/concept/firewall-filter-ex-series-overview.html`, 2021.

[84] Kalkan, K., and Alagöz, F. A distributed filtering mechanism against DDoS attacks: ScoreForCore. *Computer Networks 108* (2016).

[85] Kang, M. S., Gligor, V. D., Sekar, V., et al. Spiffy: Inducing cost-detectability tradeoffs for persistent link-flooding attacks. In *NDSS* (2016).

[86] Kang, M. S., Lee, S. B., and Gligor, V. D. The Crossfire Attack. In *2013 IEEE Symposium on Security and Privacy* (2013).

[87] Katz-Bassett, E., Madhyastha, H. V., Adhikari, V. K., Scott, C., Sherry, J., Van Wesep, P., Anderson, T. E., and Krishnamurthy, A. Reverse Traceroute. In *USENIX Symposium on Networked Systems Design and Implementation* (2010), vol. 10.

[88] Ke, Y.-M., Chen, C.-W., Hsiao, H.-C., Perrig, A., and Sekar, V. CICADAS: Congesting the Internet with Coordinated and Decentralized Pulsating Attacks. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security* (2016), ASIA CCS '16.

[89] Keromytis, A. D., Misra, V., and Rubenstein, D. SOS: An Architecture for Mitigating DDoS Attacks. *IEEE Journal on Selected Areas in Communications 22*, 1 (Jan 2004).

[90] Khattab, S., Gobriel, S., Melhem, R., and Mosse, D. Live Baiting for Service-Level DoS Attackers. In *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications* (2008).

[91] Killalea, T. Recommended internet service provider security services and procedures. Tech. rep., 2000.

[92] Koch, T., Katz-Bassett, E., Heidemann, J., Calder, M., Ardi, C., and Li, K. Anycast in context: A tale of two systems. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference* (2021), SIGCOMM '21.

[93] Koch, T., Katz-Bassett, E., Heidemann, J., Calder, M., Ardi, C., and Li, K. Anycast in context: A tale of two systems. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference* (2021), SIGCOMM '21, Association for Computing Machinery.

[94] Kohler, E., Morris, R., Chen, B., Jannotti, J., and Kaashoek, M. F. The Click Modular Router. *ACM Trans. Comput. Syst. 18*, 3 (Aug. 2000).

[95] Kokulu, F. B., Soneji, A., Bao, T., Shoshitaishvili, Y., Zhao, Z., Doupé, A., and Ahn, G.-J. Matched and Mismatched SOCs: A Qualitative Study on Security Operations Center Issues. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)* (2019).

[96] KrebsOnSecurity. KrebsOnSecurity Hit With Record DDoS. `https://krebsonsecurity.com/2016/09/krebsonsecurity-hit-with-record-ddos`, 2016.

[97] Kumari, W., and McPherson, D. Remote Triggered Black Hole Filteringwith Unicast Reverse Path Forwarding (uRPF). Tech. rep., 2009.

[98] Kupreev, O., Badovskaya, E., and Gutnikov, A. DDoS attacks in Q4 2020. `https://securelist.com/ddos-attacks-in-q4-2020/100650/`, 2020.

[99] Kuzmanovic, A., and Knightly, E. W. Low-rate tcp-targeted denial of service attacks: The shrew vs. the mice and elephants. In *Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (2003), SIGCOMM '03.

[100] Labovitz, C. The world at risk: DDoS in 2021 with Dr. Craig Labovitz (Nokia Deepfield). `https://www.youtube.com/watch?v=xnduIFdRHxk`, 2021.

[101] Lakhina, A., Crovella, M., and Diot, C. Characterization of Network-Wide Anomalies in Traffic Flows. In *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement* (2004), IMC '04.

[102] Lantz, B., Heller, B., and McKeown, N. A Network in a Laptop: Rapid Prototyping for Software-Defined Networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks* (2010), Hotnets-IX.

[103] Lee, S. B., Kang, M. S., and Gligor, V. D. CoDef: Collaborative Defense against Large-Scale Link-Flooding Attacks. In *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies* (2013), CoNEXT '13.

[104] Leis, V., Kemper, A., and Neumann, T. The Adaptive Radix Tree: ARTful Indexing for Main-Memory Databases. In *IEEE 29th International Conference on Data Engineering* (2013).

[105] Li, J., Sung, M., Xu, J., and Li, L. Large-Scale IP Traceback in High-Speed Internet: Practical Techniques and Theoretical Foundation. In *IEEE Symposium on Security and Privacy* (2004).

[106] Li, X., Bian, F., Crovella, M., Diot, C., Govindan, R., Iannaccone, G., and Lakhina, A. Detection and Identification of Network Anomalies Using Sketch Subspaces. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement* (2006), IMC '06.

[107] Liu, X., Li, A., Yang, X., and Wetherall, D. Passport: Secure and adoptable source authentication. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation* (2008), NSDI'08.

[108] Liu, X., Yang, X., and Lu, Y. To Filter or to Authorize: Network-Layer DoS Defense against Multimillion-Node Botnets. *SIGCOMM Computer Communication Review 38*, 4 (2008).

[109] Liu, Z., Jin, H., Hu, Y.-C., and Bailey, M. MiddlePolice: Toward Enforcing Destination-Defined Policies in the Middle of the Internet. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (2016), CCS '16.

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

[111] Luckie, M., Beverly, R., Koga, R., Keys, K., Kroll, J. A., and Claffy, K. Network hygiene, incentives, and regulation: deployment of source address validation in the internet. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (CCS)* (2019).

[112] Luminati. World's Largest Proxy Service. `https://luminati.io`, 2019.

[113] Luo, X., and Chang, R. K. On a New Class of Pulsing Denial-of-Service Attacks and the Defense. In *Proceedings of the Network and Distributed System Security Symposium* (2005).

[114] Mahajan, R., Wetherall, D., and Anderson, T. Understanding BGP Misconfiguration. In *ACM SIGCOMM Computer Communication Review* (2002).

[115] Mahimkar, A., Dange, J., Shmatikov, V., Vin, H., and Zhang, Y. dFence: Transparent Network-based Denial of Service Mitigation. *4th USENIX Symposium on Networked Systems Design & Implementation* (2007).

[116] Majkowski, M. The real cause of large DDoS - IP Spoofing. `https://blog.cloudflare.com/the-root-cause-of-large-ddos-ip-spoofing/`, 2018.

[117] man7.org. cgroups - Linux control groups. `http://man7.org/linux/man-pages/man7/cgroups.7.html`, 2020.

[118] man7.org. ip-netns - process network namespace management. `http://man7.org/linux/man-pages/man8/ip-netns.8.html`, 2020.

[119] man7.org. namespaces - overview of Linux namespaces. `http://man7.org/linux/man-pages/man7/namespaces.7.html`, 2020.

[120] man7.org. tc - show / manipulate traffic control settings. `http://man7.org/linux/man-pages/man8/tc.8.html`, 2020.

[121] MANRS. MANRS for Network Operators. `https://www.manrs.org/isps`, 2022.

[122] Mao, Z. M., Qiu, L., Wang, J., and Zhang, Y. On AS-Level Path Inference. In *ACM SIGMETRICS* (2005).

[123] Marques, P., Sheth, N., Raszuk, R., Greene, B., Mauch, J., and McPherson, D. Dissemination of Flow Specification Rules. Tech. rep., 2009.

[124] Memcached. Memcached — releasenotes156. `https://github.com/memcached/memcached/wiki/ReleaseNotes156`, 2018.

[125] Mergendahl, S., and Li, J. Rapid: Robust and adaptive detection of distributed denial-of-service traffic from the internet of things. In *IEEE Conference on Communications and Network Security (CNS)* (2020).

[126] Mergendahl, S., Sisodia, D., Li, J., and Cam, H. FR-WARD: Fast Retransmit as a Wary but Ample Response to Distributed Denial-of-Service Attacks from the Internet of Things. In *2018 27th International Conference on Computer Communication and Networks (ICCCN)* (2018).

[127] Meyer, D. University of Oregon Route Views Project. *University of Oregon* (2001).

[128] Miano, S., Bertrone, M., Risso, F., Bernal, M. V., Lu, Y., and Pi, J. Securing linux with a faster and scalable iptables. *SIGCOMM Computer Communication Review 49*, 3 (2019).

[129] Michael Oliveira. Hacking group says they do it for the 'lulz'. `https://www.theglobeandmail.com/technology/tech-news/hacking-group-says-they-do-it-for-the-lulz/article556865/`, 2011.

[130] Mirkovic, J., Kline, E., and Reiher, P. Resect: Self-learning traffic filters for ip spoofing defense. In *Proceedings of the 33rd Annual Computer Security Applications Conference* (2017), ACSAC '17.

[131] Mirkovic, J., Prier, G., and Reiher, P. Attacking DDoS at the Source. In *IEEE International Conference on Network Protocols* (2002).

[132] Morein, W. G., Stavrou, A., Cook, D. L., Keromytis, A. D., Misra, V., and Rubenstein, D. Using Graphic Turing Tests to Counter Automated DDoS Attacks against Web Servers. In *Proceedings of the 10th ACM Conference on Computer and Communications Security* (2003), CCS '03.

[133] Morrison, D. R. PATRICIA—Practical Algorithm To Retrieve Information Coded in Alphanumeric. *Journal of the ACM 15* (1968).

[134] Moura, G. C., Hesselman, C., Schaapman, G., Boerman, N., and de Weerdt, O. Into the ddos maelstrom: a longitudinal study of a scrubbing service. *IEEE European Symposium on Security and Privacy Workshops* (2020).

[135] Moura, G. C., Schmidt, R. d. O., Heidemann, J., de Vries, W. B., Muller, M., Wei, L., and Hesselman, C. Anycast vs. ddos: Evaluating the november 2015 root dns event. In *Proceedings of the 2016 Internet Measurement Conference* (2016), IMC '16.

[136] Murugesan, V., Selvaraj, M. S., and Yang, M.-H. HPSIPT: A High-Precision Single-Packet IP Traceback Scheme. *Computer Networks 143* (2018).

[137] Nawrocki, M., Blendin, J., Dietzel, C., Schmidt, T. C., and Wählisch, M. Down the black hole: Dismantling operational practices of bgp blackholing at ixps. In *Proceedings of the Internet Measurement Conference* (2019), IMC '19.

[138] NETCOUT. Arbor Cloud DDoS Protection Services. `https://www.netscout.com/product/arbor-cloud`, 2021.

[139] NETSCOUT. NETSCOUT's 14th Annual Worldwide Infrastructure Security Report. `https://www.netscout.com/report`, 2020.

[140] Netscout. Issue 6: Findings from 2H 2020 — Netscout Threat Intelligence Report. `https://web.archive.org/web/20210427130130/https://www.netscout.com/threatreport`, 2021.

[141] Neustar. UltraDDoS Protect Mitigation Service. `https://www.home.neustar/resources/product-literature/ddos-mitigation-service-product-literature`, 2021.

[142] Nippon Telegraph and Telephone Corporation. Welcome to RYU the network operating system (NOS). `https://ryu.readthedocs.io/en/latest/`, 2021.

[143] ntp.org. April 2010: DRDoS / Amplification Attack using ntpdc monlist command. `http://support.ntp.org/bin/view/Main/SecurityNotice`, 2010.

[144] Nur, A. Y., and Tozal, M. E. Record Route IP Traceback: Combating DoS Attacks and the Variants. *Computers & Security 72* (2018).

[145] Nvidia. Cumulus Linux 4.2 User Guide. `https://web.archive.org/web/20220124023544/https://docs.nvidia.com/networking-ethernet-software/cumulus-linux-42/System-Configuration/Netfilter-ACLs/#hardware-limitations-on-number-of-rules`, 2022.

[146] Nychis, G., Sekar, V., Andersen, D. G., Kim, H., and Zhang, H. An empirical evaluation of entropy-based traffic anomaly detection. In *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement* (2008), IMC '08.

[147] Oikonomou, G., Mirkovic, J., Reiher, P., and Robinson, M. A Framework for a Collaborative DDoS Defense. In *Proceedings of the 22nd Annual Computer Security Applications Conference* (2006), ACSAC '06.

[148] Ookla. Speedtest by Ookla Global Fixed and Mobile Network Performance Map Tiles. `https://github.com/teamookla/ookla-open-data`, 2021.

[149] Opensignal. The State of Mobile Network Experience. `https://www.opensignal.com/sites/opensignal-com/files/data/reports/global/data-2019-05/the_state_of_mobile_experience_may_2019_0.pdf`, 2019.

[150] Osanaiye, O., Cai, H., Choo, K.-K. R., Dehghantanha, A., Xu, Z., and Dlodlo, M. Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing. *EURASIP Journal on Wireless Communications and Networking*, 1 (May 2016).

[151] Papadopoulos, C., Lindell, R., Mehringer, J., Hussain, A., and Govindan, R. COSSACK: Coordinated Suppression of Simultaneous Attacks. In *Proceedings DARPA Information Survivability Conference and Exposition* (April 2003), vol. 1.

[152] Park, J., Nyang, D., and Mohaisen, A. Timing is almost everything: Realistic evaluation of the very short intermittent ddos attacks. In *Annual Conference on Privacy, Security and Trust (PST)* (2018).

[153] Park, K., and Lee, H. On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets. *SIGCOMM Computer Communication Review 31*, 4 (2001).

[154] Patel, H., and Jinwala, D. C. LPM: A Lightweight Authenticated Packet Marking Approach for IP Traceback. *Computer Networks 140* (2018).

[155] PeeringDB. Cogent Communications, Inc. `https://www.peeringdb.com/net/267`, 2021.

[156] Peterson, B. Secure Network Design: Micro Segmentation. `https://www.sans.org/reading-room/whitepapers/bestprac/secure-network-design-micro-segmentation-36775`, 2016.

[157] Peuster, M., Karl, H., and van Rossem, S. MeDICINE: Rapid prototyping of production-ready network services in multi-PoP environments. In *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)* (2016).

[158] Prasad, R., Dovrolis, C., Murray, M., and Claffy, K. Bandwidth estimation: metrics, measurement techniques, and tools. *IEEE Network 17*, 6 (2003).

[159] Prince, M. A Brief Primer on Anycast. `https://blog.cloudflare.com/a-brief-anycast-primer/`, 2011.

[160] Radware. Cloud DDoS Protection Service. `https://www.radware.com/products/cloud-ddos-services2`, 2021.

[161] Ramanathan, S., Mirkovic, J., Yu, M., and Zhang, Y. SENSS Against Volumetric DDoS Attacks. In *Proceedings of the 34th Annual Computer Security Applications Conference* (2018), ACSAC'18.

[162] Ranjan, S., Swaminathan, R., Uysal, M., and Knightly, E. DDoS-Resilient Scheduling to Counter Application Layer Attacks Under Imperfect Detection. In *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications* (April 2006).

[163] Ranjan, S., Swaminathan, R., Uysal, M., Nucci, A., and Knightly, E. DDoS-Shield: DDoS-Resilient Scheduling to Counter Application Layer Attacks. *IEEE/ACM Transactions on Networking 17*, 1 (Feb 2009).

[164] Rasti, R., Murthy, M., Weaver, N., and Paxson, V. Temporal lensing and its application in pulsing denial-of-service attacks. In *IEEE Symposium on Security and Privacy* (2015).

[165] Rasti, R., Murthy, M., Weaver, N., and Paxson, V. Temporal lensing and its application in pulsing denial-of-service attacks. In *2015 IEEE Symposium on Security and Privacy* (2015).

[166] Red Hat. TCP SACK PANIC - Kernel vulnerabilities - CVE-2019-11477, CVE-2019-11478 & CVE-2019-11479. `https://access.redhat.com/security/vulnerabilities/tcpsack`, 2019.

[167] RSnake. Slowloris HTTP DoS. `https://web.archive.org/web/20150426090206/http://ha.ckers.org/slowloris`, 2009.

[168] Santanna, J., van Rijswijk-Deij, R., Hofstede, R., Sperotto, A., Wierbosch, M., Zambenedetti Granville, L., and Pras, A. Booters - An Analysis of DDoS-as-a-Service Attacks. In *IFIP/IEEE International Symposium on Integrated Network Management* (2015).

[169] Savage, S., Wetherall, D., Karlin, A., and Anderson, T. Practical Network Support for IP Traceback. In *ACM SIGCOMM* (2000).

[170] Schimmel, I. Mellanox OVS TC. `https://github.com/Mellanox/mlxsw/wiki/OVS`, 2019.

[171] Schroeder, B., Wierman, A., and Harchol-Balter, M. Open versus closed: A cautionary tale. In *3rd Symposium on Networked Systems Design and Implementation (NSDI)* (2006), USENIX.

[172] Sekar, V., Duffield, N., Spatscheck, O., van der Merwe, J., and Zhang, H. LADS: Large-scale Automated DDoS Detection System. *USENIX Annual Technical Conference, General Track* (2006).

[173] Sermpezis, P., and Kotronis, V. Inferring catchment in internet routing. *Proceedings of the ACM on Measurement and Analysis of Computing Systems 3*, 2 (2019).

[174] ServerHunter. Find a server. `https://www.serverhunter.com`, 2019.

[175] Shan, H., Wang, Q., and Pu, C. Tail Attacks on Web Applications. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)* (2017).

[176] Shan, Y., Kesidis, G., and Fleck, D. Cloud-side shuffling defenses against ddos attacks on proxied multiserver systems. In *Proceedings of the 2017 on Cloud Computing Security Workshop* (2017), CCSW '17.

[177] Shi, L., Li, J., Zhang, M., and Reiher, P. On Capturing DDoS Traffic Footprints on the Internet. *IEEE Transactions on Dependable and Secure Computing* (2021).

[178] Shi, L., Mergendahl, S., Sisodia, D., and Li, J. Bridging missing gaps in evaluating DDoS research. In *13th USENIX Workshop on Cyber Security Experimentation and Test (CSET 20)* (2020).

[179] Shi, L., Sisodia, D., Zhang, M., Li, J., Dainotti, A., and Reiher, P. The Catch-22 Attack (Extended Abstract). In *Annual Computer Security Applications Conference (ACSAC)* (2019).

[180] Shi, L., Zhang, M., Li, J., and Reiher, P. PathFinder: Capturing DDoS Traffic Footprints on the Internet. In *2018 IFIP Networking Conference (IFIP Networking) and Workshops* (2018), IEEE.

[181] Silveira, F., Diot, C., Taft, N., and Govindan, R. ASTUTE: Detecting a Different Class of Traffic Anomalies. *SIGCOMM Computer Communication Review 40*, 4 (2010).

[182] Simon, J. Now Available: New C5d Instance Sizes and Bare Metal Instances. `https://aws.amazon.com/blogs/aws/now-available-new-c5d-instance-sizes-and-bare-metal-instances/`, 2019.

[183] SimpleWeb.org. Traces. `https://www.simpleweb.org/wiki/index.php/Traces`, 2015.

[184] Singh, K., Singh, P., and Kumar, K. A Systematic Review of IP Traceback Schemes for Denial of Service Attacks. *Computers & Security 56* (2016).

[185] Sisodia, D., Li, J., and Jiao, L. In-Network Filtering of Distributed Denial-of-Service Trafficwith Near-Optimal Rule Selection. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security* (2020), AsiaCCS '20.

[186] Smith, J. M., and Schuchard, M. Routing Around Congestion: Defeating DDoS Attacks and Adverse Network Conditions via Reactive BGP Routing. *IEEE Symposium on Security and Privacy* (2018).

[187] Snoeren, A. C., Partridge, C., Sanchez, L. A., Jones, C. E., Tchakountio, F., Kent, S. T., and Strayer, W. T. Hash-Based IP Traceback. In *ACM SIGCOMM* (2001).

[188] Soldo, F., Argyraki, K., and Markopoulou, A. Optimal Source-Based Filtering of Malicious Traffic. *IEEE/ACM Transactions on Networking 20*, 2 (April 2012).

[189] Soldo, F., Markopoulou, A., and Argyraki, K. Optimal Filtering of Source Address Prefixes: Models and Algorithms. In *IEEE INFOCOM 2009* (April 2009).

[190] Sommese, R., Bertholdo, L., Akiwate, G., Jonker, M., van Rijswijk-Deij, R., Dainotti, A., Claffy, K., and Sperotto, A. MAnycast2: Using Anycast to Measure Anycast. In *Proceedings of the ACM Internet Measurement Conference* (2020), IMC '20.

[191] Soule, A., Salamatian, K., and Taft, N. Combining filtering and statistical methods for anomaly detection. In *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement* (2005), IMC '05.

[192] Spatscheck, O., and Peterson, L. L. Defending Against Denial of Service Attacks in Scout. In *Proceedings of the 3rd Symposium on Operating Systems Design and Implementation (OSDI'99)* (1999).

[193] Stavrou, A., and Keromytis, A. D. Countering dos attacks with stateless multipath overlays. In *Proceedings of the 12th ACM Conference on Computer and Communications Security* (2005), CCS '05.

[194] Stone, R. Centertrack: An IP Overlay Network for Tracking DoS Floods. In *Proceedings of the 9th Conference on USENIX Security Symposium* (2000), p. 15.

[195] Studer, A., and Perrig, A. The Coremelt Attack. In *Computer Security – ESORICS 2009* (2009), pp. 37–52.

[196] Sun, H., Lui, J. C. S., and Yau, D. K. Y. Defending against low-rate tcp attacks: Dynamic detection and protection. In *Proceedings of the 12th IEEE International Conference on Network Protocols* (2004), ICNP '04.

[197] System, N. C. A. DNS amplification attacks. `https://us-cert.cisa.gov/ncas/alerts/TA13-088A`, 2013.

[198] The Open Networking Foundation. Introduction: What is onos? `https://wiki.onosproject.org/display/ONOS/ONOS`, 2021.

[199] Traer, S., and Bednar, P. Motives behind ddos attacks. In *Digital Transformation and Human Behavior*. Springer, 2021, pp. 135–147.

[200] Tran, M., Kang, M. S., Hsiao, H.-C., Chiang, W.-H., Tung, S.-P., and Wang, Y.-S. On the feasibility of rerouting-based ddos defenses. In *IEEE Symposium on Security and Privacy* (2019).

[201] Trusted Computing Group. Trusted platform module main specification. `https://trustedcomputinggroup.org/resource/tpm-main-specification`, 2011.

[202] Turk, D. Configuring bgp to block denial-of-service attacks. Tech. rep., 2004.

[203] Vahdat, A., Yocum, K., Walsh, K., Mahadevan, P., Kostić, D., Chase, J., and Becker, D. Scalability and Accuracy in a Large-Scale Network Emulator. *SIGOPS Oper. Syst. Rev. 36* (Dec. 2003).

[204] Venkatesan, S., Albanese, M., Amin, K., Jajodia, S., and Wright, M. A moving target defense approach to mitigate ddos attacks against proxy-based architectures. In *IEEE Conference on Communications and Network Security (CNS)* (2016).

[205] Vishwanath, K. V., and Vahdat, A. Evaluating distributed systems: Does background traffic matter? In *USENIX Annual Technical Conference* (2008).

[206] von Ahn, L., Blum, M., Hopper, N. J., and Langford, J. Captcha: Using hard ai problems for security. In *Advances in Cryptology — EUROCRYPT 2003* (2003).

[207] Walfish, M., Vutukuru, M., Balakrishnan, H., Karger, D., and Shenker, S. DDoS Defense by Offense. *ACM Transactions on Computer Systems 28*, 1 (2010).

[208] Wang, A., Chang, W., Chen, S., and Mohaisen, A. Delving Into Internet DDoS Attacks by Botnets: Characterization and Analysis. *IEEE/ACM Transactions on Networking 26*, 6 (2018).

[209] Wang, A., Chang, W., Chen, S., and Mohaisen, A. A Data-Driven Study of DDoS Attacks and Their Dynamics. *IEEE Transactions on Dependable and Secure Computing 17*, 3 (2020).

[210] Wang, H., Xu, L., and Gu, G. FloodGuard: A DoS Attack Prevention Extension in Software-Defined Networks. In *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks* (June 2015).

[211] Wette, P., Dräxler, M., Schwabe, A., Wallaschek, F., Zahraee, M. H., and Karl, H. Maxinet: Distributed emulation of software-defined networks. In *IFIP Networking Conference* (2014).

[212] Xiang, Y., Li, K., and Zhou, W. Low-Rate DDoS Attacks Detection and Traceback by Using New Information Metrics. *IEEE Transactions on Information Forensics and Security 6*, 2 (2011).

[213] Xu, Y., and Liu, Y. DDoS Attack Detection Under SDN Context. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications* (2016).

[214] Yaar, A., Perrig, A., and Song, D. SIFF: A Stateless Internet Flow Filterto Mitigate DDoS Flooding Attacks. In *IEEE Symposium on Security and Privacy* (May 2004).

[215] Yaar, A., Perrig, A., and Song, D. FIT: Fast Internet Traceback. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies* (2005), vol. 2.

[216] Yan, Q., Yu, F. R., Gong, Q., and Li, J. Software-Defined Networking (SDN) and Distributed Denial of Service (DDoS) Attacks in Cloud Computing Environments: A Survey, Some Research Issues, and Challenges. *IEEE Communications Surveys and Tutorials 18*, 1 (2016).

[217] Yi Xie, and Shun-Zheng Yu. Monitoring the application-layer ddos attacks for popular websites. *IEEE/ACM Transactions on Networking 17*, 1 (Feb 2009).

[218] Yoachimik, O. Cloudflare thwarts 17.2m rps ddos attack — the largest ever reported. `https://blog.cloudflare.com/cloudflare-thwarts-17-2m-rps-ddos-attack-the-largest-ever-reported/`, 2021.

[219] Yoachimik, O., and Ganti, V. Network-layer DDoS attack trends for Q3 2020. `https://blog.cloudflare.com/network-layer-ddos-attack-trends-for-q3-2020/`, 2020.

[220] Yoon, M., Li, T., Chen, S., and Peir, J.-K. Fit a Compact Spread Estimator in Small High-Speed Memory. *IEEE/ACM Transactions on Networking 19*, 5 (2011).

[221] York, K. Dyn Statement on 10/21/2016 DDoS Attack. `http://dyn.com/blog/dyn-statement-on-10212016-ddos-attack`, 2016.

[222] You, W., Jiao, L., Li, J., and Zhou, R. Scheduling DDoS Cloud Scrubbing in ISP Networks via Randomized Online Auctions. In *IEEE INFOCOM 2020 - The 39th Annual IEEE International Conference on Computer Communications* (2020).

[223] Yu, M., Jose, L., and Miao, R. Software defined traffic measurement with opensketch. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation* (2013), NSDI'13.

[224] Zhang, C., Cai, Z., Chen, W., Luo, X., and Yin, J. Flow level detection and filtering of low-rate ddos. *Computer Networks 56*, 15 (2012).

[225] Zhang, M., Li, G., Wang, S., Liu, C., Chen, A., Hu, H., Gu, G., Li, Q., Xu, M., and Wu, J. Poseidon: Mitigating volumetric ddos attacks with programmable switches. In *Proceedings of NDSS* (2020).

[226] Zhang, M., Shi, L., Sisodia, D., Li, J., and Reiher, P. On Multi-Point, In-Network Filtering of Distributed Denial-of-Service Traffic. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)* (April 2019).

[227] Zhang, X., Hsiao, H.-C., Hasker, G., Chan, H., Perrig, A., and Andersen, D. G. SCION: Scalability, Control, and Isolation on Next-Generation Networks. In *IEEE Symposium on Security and Privacy* (2011).

[228] Zhang, X., Sen, T., Zhang, Z., April, T., Chandrasekaran, B., Choffnes, D., Maggs, B. M., Shen, H., Sitaraman, R. K., and Yang, X. Anyopt: Predicting and optimizing ip anycast performance. In *Proceedings of the 2021 ACM SIGCOMM 2021 Conference* (2021), SIGCOMM '21.

[229] Zheng, J., Li, Q., Gu, G., Cao, J., Yau, D. K., and Wu, J. Realtime DDoS Defense Using COTS SDN Switches via Adaptive Correlation Analysis. *IEEE Transactions on Information Forensics and Security 13*, 7 (July 2018).