Two Decades of DDoS Attacks and Defenses

LUMIN SHI, University of Oregon

Two decades after the first distributed denial-of-service (DDoS) attack, the Internet continues to face DDoS attacks. To understand why DDoS defense is a difficult problem, we must study how the attacks are carried out and whether the existing defense solutions are sufficient. In this work, we review the latest DDoS attacks and DDoS defense solutions. In particular, we focus on the key advancements and missing pieces in DDoS research.

CCS Concepts: • **Networks** → **Network security**; **Denial-of-service attacks**.

Additional Key Words and Phrases: DDoS attack, DDoS defense, CDN, SDN

1 INTRODUCTION

Distributed denial-of-service (DDoS) attacks are easy to launch but difficult to defend against. Despite two decades of DDoS research, DDoS attacks continue to impose threats to today's networks. Recent attacks such as [1, 2, 3] have shown that resourceful attackers can launch DDoS attacks to flood network links and leaves them paralyzed for hours or days. In the meantime, the proliferation of Internet-of-Things (IoT) devices amplifies the threat of DDoS. Many of these devices come with little to no updates for security vulnerabilities, and as such, they are exploitable by the attackers. Compromised IoT devices remain vulnerable even after major incidents such as the Mirai attack [1]. The sheer amount of vulnerable, increasing IoT devices challenges the capacity/scalability of existing defense solutions.

On the other hand, the frequent DDoS attacks led the network community to seek better defense solutions. From remotely triggered black hole (RTBH) [4] to BGP FlowSpec [5], we see the progress made by the network community to move away from high-collateral-damage solutions [4]. The community is paying more attention to deploy fine-grained traffic filters [5]. DDoS protection service (DPS) providers such as Akamai and Cloudflare utilize anycast techniques to redirect traffic of the attacked networks to their data centers and then defend against the attacks using their proprietary solutions [6]. The research community continues to design and optimize defense solutions. At the time of this writing, there are hundreds of projects that focus on different defense strategies [7, 8, 9, 10], i.e., attack *prevention*, *detection/classification*, *mitigation*, and *attribution*. Yet many promising solutions are not being widely deployed years after the publication [11, 12, 13]. For this reason, many recent research projects pay for extra attention to deployability issues [14, 15, 16]. Meanwhile, network security researchers are also exploring more advanced attack methods [17, 18, 19, 20] that challenge yet-to-be-deployed defense solutions.

The goal of this report is to help us and the readers to understand the latest DDoS attacks and defense solutions better, and more importantly, to find the missing pieces from the previous research. We evaluate each attack/defense project with the following aspects: 1) the problems, 2) the assumptions, and 3) the techniques. We organize the remaining report as follows: in Sec. 2, we introduce and classify the basics of known DDoS attacks and defense solutions from a topological perspective. In Sec. 3, we study the latest DDoS attacks. We later study the existing DDoS defense solutions in Sec. 4. We further divide Sec. 4 into three sub-sections based their defense strategies: attack prevention solutions in Sec. 4.1, detection/classification solutions in Sec. 4.2 and mitigation solutions in Sec. 4.3. We then summarize previous DDoS surveys in Sec. 5. Finally, in Sec. 6, we 1) conclude the report, 2) point out the missing pieces in DDoS research, and 3) briefly present an ongoing DDoS attack project that studies the fundamental limitations of the DDoS defense today.

Author's address: Lumin Shi, University of Oregon, luminshi@cs.uoregon.edu.

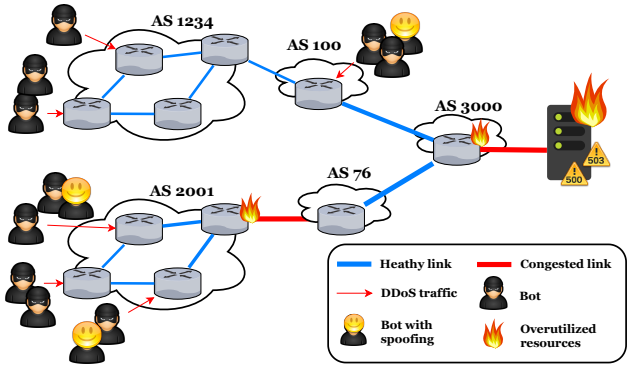

Fig. 1. DDoS attacks locations

In this report, we selected and reviewed roughly 100 papers in this report, and most of them are from the following venues:

- *Conference*: SIGCOMM, USENIX (ATC, Security, NSDI, OSDI), IEEE S&P, IMC, INFOCOM, ACSAC, CoNext, ICNP, NDSS, ACM CCS, ACM ASIACCS, and ESORICS.
- *Journal*: ToN, CCR, and Computer Networks.

2 DDOS ATTACK AND DEFENSE PRIMER

In this section, we first introduce the resources that DDoS attacks can target (Sec. 2.1), our taxonomy of the DDoS attacks (Sec. 2.2), and how attackers prepare their attacks (Sec. 2.3). We then cover our taxonomy of DDoS defenses (Sec. 2.4), as well as the pros and cons of the defenses at different locations of the Internet.

2.1 Attack Targets

The Internet allows any network to communicate with any other network, and this property fundamentally enables DDoS attacks. The adversaries can exploit vulnerable devices, launch DDoS attacks to and from these devices, and even mask the attack traffic as benign traffic. The primary goal of a DDoS attack is to overwhelm specific resource that affects the regular operation of its victim(s). The resource includes but is not limited to the network processors of hardware routers, the memory and input/output (I/O) resource of a web server, and the bandwidth of a link. Figure 1 illustrates that vulnerable resources can be anywhere on the Internet.

2.1.1 Edge Networks. Commonly, DDoS attacks are launched against edge networks for two reasons: 1) the inbound link bandwidth of an edge network is much lower than a link in a transit network, 2) edge networks contain more applications that are vulnerable to DDoS attacks. To summarize, in an edge network, an attacker can overwhelm the following resources:

Computational resource: an end host cannot keep up with computational tasks requested by the attacker. E.g., the attack consists of frequent and crafted HTTP requests to a server, and these requests trigger the server to perform computationally heavy tasks.

Memory and input/output (I/O) resource: a crafted attack forces the victim server to perform a vast amount of I/O operations. E.g., the attacker can force a web server to maintain an excessive amount of network connections to invalidate the high-speed memory cache on the server. As a result, the server spends more time on I/O than computation.

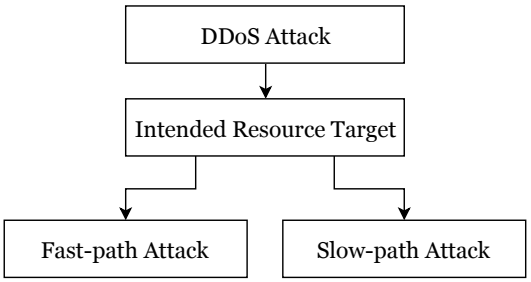

Fig. 2. DDoS attack taxonomy in this survey

Network bandwidth resource: an attacker can overwhelm the link bandwidth of a victim, be it a link from the ISP to the victim or a link inside a victim's own network.

Note that in the real world, many attacks are often a mixture of both resources above. An attacker cannot attack memory resources without invoking the computational resource. E.g., an HTTP request could trigger the victim server to perform a set of hash functions (computational resource) after the server makes several queries to its database (I/O resource).

2.1.2 Transit Networks. As we connect more devices to the Internet, some of which have serious security flaws, they enable attackers to build large-scale botnets more easily. These large-scale botnets made attacks on transit networks possible. Each link on the Internet has a finite amount of bandwidth to carry traffic. An attacker can command a botnet to send a tremendous amount of traffic towards the target link to cause link congestion. In Figure 1, we see link congestion between the autonomous system (AS) 2001 and 76 due to the overwhelming amount of attack traffic. To be more specific, AS 2001's border router can no longer cope with all the traffic towards AS 76; the border router's packet buffer is fully occupied with the DDoS packets while the network processor of the router cannot process packets any faster.

Unfortunately, links are not the only attack target in transit networks. The recent development of software-defined networking (SDN) incentivized ASes and Internet exchange points to operate their networks [21]. While SDN brings flexibility to the networks, it relies on a piece of software (SDN controller) to implement network functions such as the traditional control-plane software. Like any software, SDN creates new vulnerabilities for the attacker to exploit. For example, an attacker can exploit the controller-to-switch communication channel in SDN to launch a DoS attack [10, 22].

2.2 DDoS Attack Taxonomy

While there are many ways to classify a DDoS attack (previous surveys such as [7] proposed over eight DDoS classes), we focus on the *critical path*, i.e., *fast path* and *slow path* [23], of DDoS attacks, as shown in Figure 2. Note that in this report, we classify an attack based on its design goal: the *intended resource target* that the attack tries to overwhelm. We define fast-path and slow-path attacks as follows:

Fast-path attacks: These are attacks that directly inundate hardware resources such as router buffers and application-specific integrated circuit (ASIC) chips, with or without domain knowledge of the running software. In other words, a fast-path attack does not focus on the software that controls the hardware resources. *Common* fast-path attacks today are easier to detect than the slow-path attacks, i.e., they share discernible packet payloads and often do not follow the traffic pattern of benign network flows. Since a hardware router has a fixed packet switching capacity on

each line card, attackers can 1) send packets at their maximum transmission units (MTU) frequently to overrun the packet buffer of a router, or 2) send the smallest possible packets frequently to overrun the packet-switching capacity of a router. Lately, researchers proposed more stealthy fast-path attacks, e.g., [17, 18], that are more stealthy than the traditional ones. We discuss them in detail in Sec. 3.

Fast-path attack example — DNS amplification attack [24]: The attacker in such an attack sends public DNS servers spoofed DNS requests using the victim's IP address as the traffic source address. The DNS servers then send responses with more bytes than the spoofed requests to the victim. Thus, the attack amplifies the volume of the original attack traffic. Although the attack exploits the software of DNS servers and could degrade the DNS server performance, the attack's real intention is to congest its victim's inbound link(s). Therefore, we classify such an attack as a *fast-path* attack.

Slow-path attacks: These are attacks that overwhelm the software, computational, memory, or I/O resource of a device (which could be anywhere along the paths of DDoS traffic). Adversaries need domain knowledge to exploit the vulnerable software for such attacks. The slow-path attacks are *generally* more dynamic than the fast-path attacks, as the attacker can choose what software at which network layer to exploit. The attacker can also target software at various network locations, as we analyzed in section 2.1. Here we list the *common* slow-path attacks that exploit software at different network layers:

Layer 3: ICMP-based attacks, e.g., ping-of-death [25].
Layer 4: TCP SYN flood [26], TCP SACK panic [27].
Layer 5: Low-and-slow HTTP-based attacks [28] such as R.U.D.Y. [29].

The effects of slow-path attacks widely differ from one to another, e.g., TCP SACK panic attack only requires a few crafted packets to bring down a whole server, but may require millions of SYN packets to overwhelm a web server.

Slow-path attack example — TCP SYN flood attack [26]: The original design goal of the TCP SYN flood attack is to force an end host to maintain an excessive amount of half-open TCP connections. As a result, the end host is unable to process with new legitimate connections, and we classify TCP SYN flood attack as a *slow-path* attack. Generally speaking, most *slow-path* attacks allow attackers to hide the *raw attack power* their botnets posses and are more effective than *fast-path* attacks. With that said, an attacker with a massive botnet can certainly launch any *slow-path* attacks, e.g., TCP SYN flood attack, as a *fast-path* attack. Nevertheless, in this study, we classify an attack by its *designed intention*.

2.3 Attack Preparation

A DDoS attack often requires a botnet to overrun its target's resource or degrade the target's service level. In the past, an attacker needs to scan vulnerable end hosts and compromise vulnerable hosts for building a botnet. Today, the attackers can quickly assemble an enormous botnet on demand. There are multiple ways to assemble a botnet: 1) one can purchase bots from DoS-for-hire websites [30], 2) rent low-cost virtual machines from *hundreds* of cloud providers [31], or 3) even have access to *millions* of residential IPs from residential proxy providers [32].

An attacker can employ various attack techniques to command the botnet to attack the victims. Traditionally, the attack traffic is often discernible with some signatures in each attack packet, or the attack flows behave very differently from regular flows. Today, we see the attackers employ more advanced attacks than the brute-force attacks above. These attacks are often difficult, if not impossible, to detect in today's Internet, and we cover the latest attacks in Sec. 3.

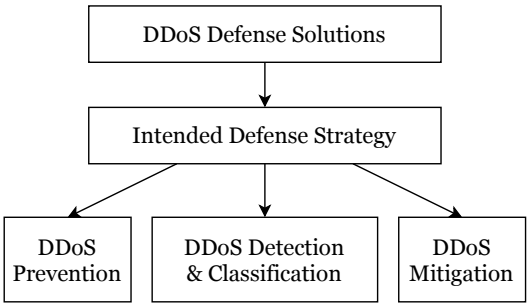

Fig. 3. DDoS defense taxonomy in this survey

Apart from the latest attacks, IP spoofing remains a prominent problem in today's Internet; by default, networks route packets based on the packet destinations, and do not validate the source address of each packet. Thus, adversaries can virtually spoof *any bit* of a packet to render most attack filtering efforts ineffective. When the affected network cannot derive valid attack signatures, it is left with mitigation solutions, such as RTBH, that negatively affect its network connectivity from and to the remaining Internet. Figure 1 shows an attack that overloaded the victim server, and the server cannot provide a normal service level to its clients. The attacker in the figure employs two attack techniques: 1) command the botnet to send discernible attack traffic to the victim network, 2) exploit IP spoofing to disrupt signature-based mitigation solutions.

Despite the network community has anti-spoofing solutions such as ingress/egress filtering [33], the number of spoofable IP networks is still at large due to the deployment rate of these anti-spoofing solutions. At the time of this writing, according to CAIDA's spoofer project [34], there are more than 23% of these ASes allow IP spoofing, and 16% of the IPv4 blocks are spoofable.

2.4 DDoS Defense Taxonomy

The DDoS is a network security problem, and like any security problem, we should study its defense solutions by defense strategies: **prevention**, **detection/classification**, and **mitigation**, as shown in Figure 3. In this section, we first introduce our taxonomy (Sec. 2.4.1 – 2.4.3), as well as the *advantages* and *disadvantages* of each defense strategy at different locations on the Internet, as shown in Figure. 4.

2.4.1 Prevention. The main objective of a prevention solution is to prevent a DDoS attack 1) from being launched by the attacker, 2) reaching its victim, or 3) exploiting limited resources of the victim. We summarize prevention solutions by their deployment locations on the Internet as follows:

Source-end prevention: A prevention solution at DDoS source, defined as source end in this report, is to 1) secure end hosts to prevent adversaries from building large botnets and 2) filter DDoS traffic proactively, such as ingress filtering. In fact, ingress filtering [33] is designed to work at such a location. However, these solutions require a high deployment rate for them to achieve their designed performance. While it is difficult to compromise the end hosts that are running the latest operating systems (OS) as they receive security patches automatically, many end hosts run OSes that receive short-term or no security patches at all [1]. For example, a significant amount of end-of-support but functioning home routers are more susceptible to become bots. Pushing security solutions to networks is also challenging, although there are fewer edge networks on the Internet than end hosts. The ingress filtering deployment rate is a prime example [34]. Other network security solutions, e.g., BGP security, are also facing the same challenge [35].

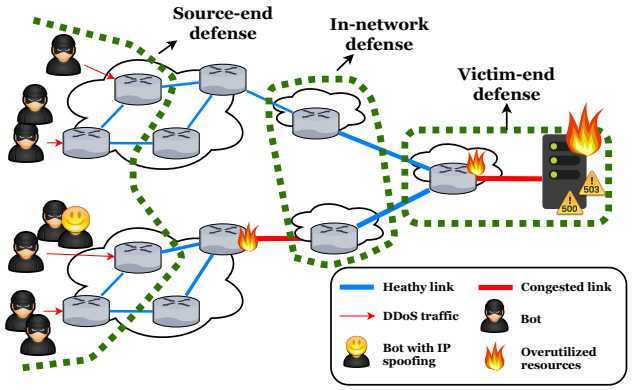

Fig. 4. DDoS defense locations

Victim-end prevention: A prevention solution from the victim end is often to 1) authenticate clients and prevent attackers from abusing resources for the clients, and 2) apply rate-limiting policies to clients based on a normal client traffic/behavior profile. Such techniques cannot stop attackers from, for example, launching fast-path attacks or exploiting unknown software vulnerabilities at the victim end.

In-network prevention: An in-network prevention solution is to 1) absorb DDoS traffic with dedicated network infrastructure, e.g., via content distribution networks (CDN), 2) dodge attacks by shifting service locations, and 3) patch/redesign the Internet architecture to become resilient against DDoS attacks. In general, an in-network prevention solution requires a lower deployment rate than a source-end prevention solution. It is inherently advantageous in traffic engineering, which enables the three prevention options above.

2.4.2 Detection/Classification. In this report, we differentiate a DDoS detection solution from a classification solution. A detection solution is to find out if there is an ongoing attack in a network. In contrast, a classification solution provides a fine-grained view of the attack, e.g., to identify attack flows by n-tuple packet header fields. On paper, deployment locations are *irrelevant* to detection/classification solutions, as long as *sufficient* telemetry information is available to the detection/classification solutions. Realistically speaking, given the deployment location of a DDoS defense system, the lack of telemetry data *limits* the types of DDoS attacks a detection/classification solution can recognize. Hence, we also divide detection/classification solutions by their intended deployment locations as follows:

Source-end detection/classification: A source-end detection/classification solution, by default, is inherently limited by the amount of available network telemetry data. While such a solution is highly scalable since the total computation load is distributed to all edge networks, it does not see the big picture as what the DDoS victim sees. Therefore, detection/classification solutions at source end are best suited for detecting/classifying significant outliers, e.g., TCP SYN flood attack.

Victim-end detection/classification: Detection/classification solutions at the victim-end location have an *inherent* advantage: a more comprehensive view of the DDoS traffic than other locations without collaborations. What is more, a victim-end solution has a higher likelihood to access end-host resource utilization information for more accurate detection/classification results. In general, a victim-end solution can access more types of telemetry data to detect various fast/slow-path attacks.

Table 1. Defense strategy suitability at different network locations

Defense strategy	Source-end	Victim-end	In-network
Prevention	High	Low	Medium
Detection/classification	Low	High	Medium
Mitigation	High	Medium	High

In-network detection/classification: In-network detection/classification solutions share the (dis)advantages of source-end and victim-end solutions but to a lesser degree. For example, an in-network solution, on average, have a better view of the victim's traffic than a source-end solution but is less scalable as it needs to process more traffic.

An in-network detection/classification solution is in a *unique* position to capture advanced attacks that may not be visible at the victim end, as we covered in Sec. 3.1, although it is unknown how motivated the transit networks are to detect the advanced attacks.

2.4.3 Mitigation. A DDoS mitigation solution in this report is about lessening the attack traffic towards the victim, i.e., reducing DDoS traffic at different Internet locations. DDoS mitigation solutions range from systems that filter DDoS traffic at different locations on the Internet to specific ones that optimize the mitigation efficiency, e.g., techniques that generate fewer traffic filters to cover more DDoS traffic or pinpoint the best location(s) to deploy traffic filters. We also divide mitigation solutions by their deployment locations as follows:

Source-end mitigation: Source-end mitigation solutions in our study are mainly about 1) traffic filtering, 2) rate limiting, and 3) ask clients to fight for their fair shares of resources. Similar to source-end prevention solutions, a mitigation solution at source end incurs a low traffic policing overhead but requires a high deployment rate.

Victim-end mitigation: Unfortunately, if a victim network is experiencing a fast-path attack, the network cannot do anything to mitigate the attack if it does not control the affected fast-path resources. For slow-path attacks, the mitigation solutions are extremely specific, e.g., write a hotfix to address TCP SACK panic [27], and we consider them to be out of scope.

In-network mitigation: In-network mitigation solutions can alleviate the attack load on victims. It is often the go-to choice for mitigating fast-path attacks. Such solutions can utilize traffic redirection techniques to distribute attack traffic to multiple mitigation facilities and mitigate the attack from there. Commonly adopted solutions by the network operators, such as RTBH [4] and BGP FlowSpec [5], are in-network mitigation solutions.

2.4.4 Summary of DDoS defense. In this section, we covered our taxonomy of DDoS defense in this report, the advantages and disadvantages of all three defense locations. Finally, we summarized the *suitability* of three defense strategies, prevention, detection/classification, and mitigation, at different network locations, as shown in Table. 1.

3 DDOS ATTACKS

We have covered the common attacks in Sec. 2.2. In this section, we focus on more advanced attacks that researchers proposed in recent years. We classify the new attacks into fast-path and slow-path attacks, based on our attack taxonomy. For each attack, we analyze 1) how difficult it is to launch, 2) how stealthy the attack is for DDoS detection, and 3) how easy it is to mitigate the attack, as shown in Table 2.

3.1 Fast-path Attacks

Fast-path attacks intend to inundate the hardware resources of network devices without invoking the software component. E.g., the packet switching speed or the packet buffer of a network processor. A fast-path attack requires the adversary to control a decent amount of bots to send a high volume of traffic towards the target network. As a result, such an attack leads to link congestion at an upstream network(s) of the target network. While many fast-path attacks today employ amplification techniques to increase the scale of the attacks, we find major DDoS incidents, e.g., the Mirai attack [1], utilize compromised IoT devices to launch fast-path attacks directly. Fortunately, almost all the fast-path attacks today have discernible traffic signatures, and the network operators can use the signatures to mitigate such attacks.

In this section, we study two fast-path attacks that are less common in the real world. Note that although the fast-path attacks today strictly target router/switch packet processing or buffer capacity, we expect future DDoS attacks to target new generation hardware that offloads today's software functions, as we covered in Sec. 3.2.

3.1.1 The Coremelt Attack. Studer *et al.* [17] proposed the Coremelt attack, a fast-path attack that does not send DDoS traffic to any edge networks as its victims. The Coremelt attack exchanges "*legitimate*" traffic between only the bots themselves, i.e., all traffic flows among the bots follow congestion control. The goal of this attack is to congest targeted links in the transit networks, thus to degrade network throughput of legitimate user traffic. Due to the attack behavior, the authors claim the Coremelt attack is difficult to detect, as all bots send *wanted* traffic to each other. Furthermore, the adversary could mimic the temporal flow patterns of benign users to mask the attack. In theory, with N bots, an adversary can generate up to $O(N^2)$ pairwise connections, which could cause cascading effects on the stateful services that happen to carry the attack traffic.

This attack may not be viable in today's Internet: the authors use inferred AS-level topologies [36] and a simplified bandwidth model to evaluate the attack. The bandwidth model does not consider more realistic factors: backbone networks are well-connected, and they can use load balancing techniques, e.g., equal-cost multi-path routing (ECMP), to distribute packets across multiple links. Although the requirement of the Coremelt attack is difficult to obtain, it may be not so unrealistic soon as the Internet continues to deploy more vulnerable IoT devices.

3.1.2 The Crossfire Attack. Kang *et al.* [18] proposed another stealthy fast-path attack, the Crossfire attack. The attack is stealthy as it does not send all attack traffic towards a single network. Instead, the bots in the Crossfire attack send low-volume "*legitimate*" flows to a set of decoy servers that share a set of links that carry traffic for the intended victims. A decoy server is a server that resides in a network that is different from the intended victim network. The Crossfire attacker first identifies the persistent link(s) that can be congested by the botnet. Then, the attacker commands the bots to send attack flows towards the decoy servers. As a result, the attack inundates the link bandwidth of the intended victim network's upstream link(s). An example Crossfire attack is as shown in Figure 5. Note that during a Crossfire attack, the intended victims are unlikely to detect the attack, as they do not see the full attack landscape (flows). Not to mention that the attack flows are "*legitimate*" flows.

The Crossfire attack requires four stages to carry out a successful attack. First, the attacker commands bots to run *traceroute* to the intended victims and decoy servers to collect router-level paths. The attacker selects the persistent links that always appear in the collected paths. It then computes the links that are heavily used by both legitimate and bot traffic, and select a subset from the collected links as attack targets. Next, the attacker coordinates its bots to send evenly distributed traffic to the decoy servers via the targeted links. The aggregate attack traffic bandwidth is slightly

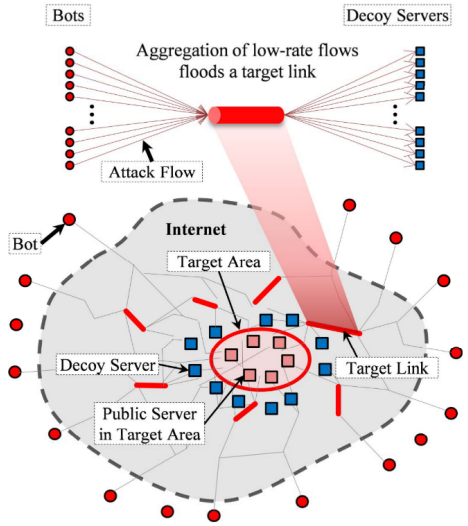

Fig. 5. The Crossfire Attack [18]

higher than the bandwidth of the targeted links, so the legitimate flows carried by the targeted links are degraded. The authors claim that since each bot only sends low-volume traffic towards the decoy servers, real-world intrusion detection systems (IDS) are unlikely to sound alarms.

When comparing the Crossfire attack to the Coremelt attack, we find the Crossfire is more realistic than the Coremelt; the Crossfire attack targets links that are closer to the edge networks and are often not over-provisioned. The Coremelt attack, on the other hand, requires a specific bot distribution on the Internet to overwhelm a particular link in a backbone network.

3.2 Slow-path Attacks

A slow-path attack exploits the running software of an end host, and it intends to either 1) trick the software to stay in a low throughput state or 2) exhaust the computational, memory, or I/O resources via specific program interfaces. To launch an effective slow-path attack against a victim, the adversary first needs to learn the vulnerable software that is running on the victim host. The adversary may also need to measure the victim's system performance/capacity to estimate the required attack budget.

In this section, we study 1) the latest pulsing attacks [19, 20] that aim to trick software to stay in a low throughput state, and 2) a DoS attack in SDN [22] that aims to exhaust the I/O resource in software-defined networks.

3.2.1 Pulsing Attacks. In a pulsing attack [37], the attacker *coordinates* the bots to send bursty traffic pulses to the victim network, periodically. These bursty traffic pulses allow the attacker to degrade the benign TCP flows' throughput that are established with the victim. A well-synchronized pulsing attack can *trick* the benign TCP flows to believe in severe link congestion. Thus, these flows reduce their TCP window size to hope for less congested links. While many pulsing attacks, such as [38, 39, 40], have been proposed, they have difficulties in 1) synchronizing the bots to launch the traffic pulses, and 2) they require a centralized controller to coordinates the bots. Therefore, it is often quite challenging to launch a successful pulsing attack. To overcome these two issues, Ke *et al.* [19] presented CICADAS, a practical pulsing attack that aims to address both issues.

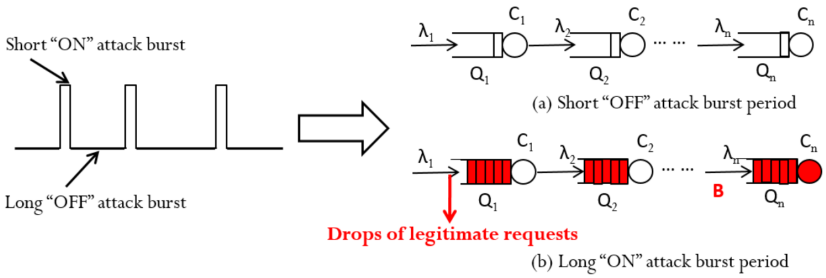

Fig. 6. Tail attack on web applications [20]

CICADAS implements 1) a decentralized attack system that uses link congestion as an implicit signaling mechanism, and 2) a Kalman-filter-based estimation algorithm to achieve tight pulse synchronization among the bots in the attack. CICADA involves the following steps to launch a pulsing attack against TCP flows:

> **Bootstrap**: the attacker first initializes each bot with a set of attack parameters, e.g., which link to attack, or how bursty the pulse should be.
> **Lurk**: once bots are initialized, each bot probes the target link to estimate the round-trip time, and infers the link congestion.
> **Attack**: once a bot detected link congestion, the bot starts to send attack pulses to the target link, and hope to congest the target link periodically. If there are enough bots to congest the target link at the same time, the bots can suppress benign TCP flows due to the TCP congestion avoidance design.

The authors of CICADAS evaluated the attack in an emulated environment, and they show when the length of each attack pulse is set to 200ms, CICADAS can effectively reduce the throughput of benign TCP flows to as low as 30%.

Shan *et al.* [20] proposed a pulsing attack against modern web applications. Authors of this work consider the attack as a low-rate DDoS attack, and they aim to force the web applications to incur long-tail latency. Unlike traditional low-rate attacks that create *long-lasting* connections with the victim servers and waste their memory or I/O resource, this attack [20] is to create *short* source contentions [41] to cause long latency (more than 1 second) on web applications at 95 percentiles. Because each source contention period is short, existing coarse-grained system telemetry tools cannot detect such an attack; on average, the attack does not overwhelm the hardware resource. Authors claim the attack can impose long-term damages to the service quality of its victims.

This attack exploits the fact that most web applications today are distributed systems; a web application is often composed of a web server, a back-end program that handles the business logic, and a database. Each component above is hosted on a separate machine for scalability concerns. The attack tries a combination of different bursty HTTP(s) requests to the victim server, and utilizes Kalman filter to estimate the most effective requests to generate predictable short source contention periods within the web application. As a result, the victim application has to drop/queue requests from legitimate users, which translates to high tail latency. Figure 6 represents the attack process. Interestingly, both CICADAS [19] and this attack [20] utilize Kalman filter to estimate the best timing for sending out attack pulses. The main difference is that the CICADAS targets the benign TCP flows (layer 4) while Shan *et al.* [20] focus on attacking web applications (layer 5).

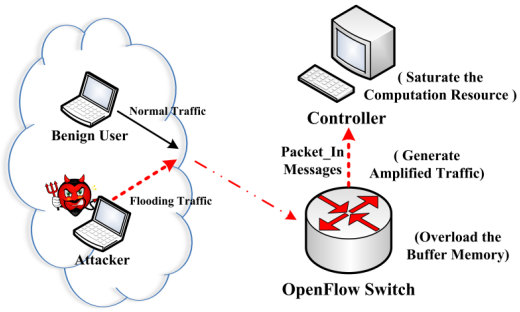

Fig. 7. DoS in SDN [22]

3.2.2 DDoS in SDN. Today, many networks employ SDN to facilitate more flexible network engineering via a common programmable interface. However, many SDN implementations, e.g., software controllers for OpenFlow, are susceptible to DDoS attacks [10].

Wang *et al.* [22] proposed a DoS attack to exhaust the I/O resource in SDN networks. This attack is viable when the target SDN network employs reactive-forwarding schemes. In a reactive-forwarding scheme, SDN switches *query* the SDN controllers to decide how to process the packets when the switches cannot match any flow entries in their forwarding tables (FIB). The switches then **cache** the decisions from SDN controllers in their FIBs and proceed to process packets. Thus, the attack is to send packets that the SDN switches do not know how to handle. If the adversary sends randomly generated packets frequently enough, an SDN switch is then bound to query its controller for learning how to process those packets. As a by-product, such an attack allows the attacker to fill up the forwarding tables of the SDN switches, which could force the switches to evacuate the benign flow entries for DDoS flow entries. In summary, the DDoS attack is to overwhelm the I/O channel between the SDN switches to their SDN controller(s), as indicated in Figure 7.

3.3 DDoS Attack Summary

In Table 2, we first list the most frequent but common DDoS attacks on the Internet today. With existing DDoS attack tools available online, these attacks are easy to launch. Most of these attacks are easy to detect and mitigate as they consist of discernible attack patterns. We rank low-rate attacks to be more difficult to detect and mitigate as their traffic profiles may overlap with the legitimate traffic profiles.

Table 2. Selected DDoS attacks

	Attack Type	Example	Attack Difficulty	Defense Difficulty	
				Detection	Mitigation
Traditional Attacks	**Fast-path**	Flooding attacks [42, 2, 24, 43]	Easy	Easy	Easy
	Slow-path	Low-rate attacks [29, 44]	Easy	Medium	Medium
		Layer 4/5 attacks [45, 46, 26]	Easy	Easy	Easy
Advanced Attacks	**Fast-path**	Coremelt [17]	Hard	Hard	Hard
		CrossFire [18]	Hard	Medium	Hard
	Slow-path	CICADAS [19]	Medium	Hard	Medium
		Tail Attack [20]	Medium	Hard	Easy
		DoS on SDN [22]	Hard	Medium	Easy

While these DDoS attacks are easy to launch and can efficiently take down many edge networks, the network community has also learned how to defend against the common DDoS over the years. For example, Handley *et al.* [47] listed general guidance to design network systems that are DDoS resilient. As many DDoS attacks today can often reach 100s of Gbps and beyond, DPS services, such as Akamai [48] and Cloudflare [49], built dedicated network infrastructure to 1) absorb DDoS traffic with DNS and BGP anycast, and 2) filter the DDoS traffic in hours if not minutes[50].

Meanwhile, network security researchers continue to discover advanced DDoS attacks, as shown in the bottom section of Table 2. In comparison, advanced DDoS attacks are more stealthy than common DDoS attacks, as we discussed in Sec. 3.1 and Sec. 3.2. While the victims can mitigate [20, 22] locally once detected, others [17, 18, 19] are more difficult to mitigate as they require mitigation efforts from the victims' upstream networks. Not to mention that a victim in such attacks needs to persuade their upstream networks to filter traffic towards other networks. Fortunately, we consider these attacks more difficult to launch as they 1) require a higher attack budget [17, 18], 2) rely on accurate measurements during the attack [19, 20], or 3) have access to specific network environments [22]. Note that as the network hardware evolves, a slow-path attack we reviewed today may become a fast-path attack in the future. We see more software functions are being offloaded to hardware [51]. Regardless, our attack taxonomy should continue to apply.

We believe our network community needs to evaluate whether the existing prevention, detection, and mitigation solutions are sufficient to defend against the advanced DDoS. To the best of our knowledge, there are no previous surveys that cover these advanced DDoS attacks, and we hope to fill this gap in our report.

4 DDOS DEFENSES

In this section, we review the DDoS defense solutions published in selected venues, as listed in Sec. 1. As DDoS attacks continue to impose threats to today's network for more than two decades, we believe there is a need for **deployable** DDoS defense research. While the following factors are by no means complete, we believe a deployable defense solution must consider them, if applicable.

Assumption: Does a solution base on unrealistic assumptions? E.g., assuming no IP spoofing.
Performance: Does the solution achieve the problem? How good is it?
Scalability: Does a solution scale against large-scale DDoS attacks?
Delay: How long does a solution take to react to a DDoS attack?
Cost: Does the cost of the solution justify its operational context?
Transparency: Does a solution require significant modifications to the Internet?
Complexity: How many moving components or dependencies does a solution require?

In particular, we focus on the **performance** and **scalability** of defense solutions, and we qualitatively evaluate the **deployability** of the solution.

Based on our defense taxonomy from Sec. 2.4, below, we cover prevention solutions (Sec. 4.1), detection/classification solutions (Sec. 4.2), and mitigation solutions (Sec. 4.3). For solutions in each defense strategy, we further divide them by the three defense locations: 1) **DDoS-source**, 2) **victim-network**, and 3) **in-network**. We summarized selected defense solutions in Tables 3, 4, and 5.

4.1 DDoS Prevention

A good defense starts with a comprehensive prevention scheme. In DDoS defense, prevention solutions are to prevent DDoS from happening. As mentioned in the section overview, we divide prevention solutions into three groups based on their deployment locations:

- DDoS-source prevention

  - secure end hosts to prevent adversaries from building large botnets, and
  - filter DDoS traffic proactively, such as ingress filtering.
- Victim-network prevention
  - authenticate clients to prevent attackers from abusing clients' resources, and
  - apply rate-limiting policy based on the normal traffic profile of each host.
- In-network prevention
  - absorb DDoS traffic with dedicated network infrastructure,
  - dodge attacks by shifting service locations, and
  - patch/redesign the Internet architecture to become resilient against DDoS attacks.

4.1.1 DDoS-Source Prevention: End-Hosts Security. As bots are the primitive elements in DDoS attacks naturally, hardening the overall system security of end-hosts on the Internet is extremely important. Generally speaking, we should always advocate the awareness of software security, e.g., enable the on-host firewall, keep the system software up to date, disable unused services, and as simple as to not execute random software downloaded from the Internet. While these practices cannot put DDoS attacks to a full stop (e.g., the adversaries can rent machines from data centers to launch DDoS), they certainly help to reduce the attack scale. These simple, probably most effective techniques, however, are often the most difficult in practice. For example, in 2016, the Mirai botnet took down Dyn's DNS service, which caused a cascading failure that renders many major web services inaccessible. Antonakakis *et al.* [1] found the root cause of the incident presents a classic example of the security oversight from many hardware manufacturers; the devices participated in this attack are publicly accessible via dictionary account and passwords, and they come with virtually no software support from day one. On top of incidents such as the Mirai attacks, researchers regularly discover design/implementation bugs in operating systems; recently, security researchers found a vulnerability in Linux TCP SACK implementation where the adversaries can easily trigger kernel panic on any vulnerable servers [27].

4.1.2 DDoS-Source Prevention: IP-Spoofing Prevention. Ingress/egress filtering technique [33, 52] is one of the most effective prevention schemes to prevent amplification attacks from happening. Although it is a simple yet effective solution, the deployment of such a technique is low in many countries [34]. Ingress/egress filtering requires the edge network operators to install stateless ACLs on the routers to rule out outgoing traffic with spoofed source IP addresses; i.e., in an edge network, the ACL only allow traffic sourced from their network prefixes. This technique requires a high deployment rate for effective spoofing prevention, and it is not ideal for transit network deployment. Often, IP-spoofing prevention solution for the transit networks means a system with inter-AS collaborations, to obtain the knowledge of expected traffic in the transit networks [53, 54].

4.1.3 Victim-Network Prevention: Authentication. As a common practice, public-facing servers should verify if a user is a human or bot before running any resource-intensive operations; Systems such as CAPTCHA [55] helps to differentiate human from the bot by asking a set of puzzles that are computationally difficult for the bots. To provide better user experience than CAPTCHA, Gummadi *et al.* [56], presented a human activity attester to authenticate human users. The attester collects and models the keyboard and mouse activity of a benign user, and rejects requests that do not follow the model. The work relies on Trusted Platform Module [57] to ensure the authenticity of the activities from input devices, thus to can prevent attackers from fabricating activity data.

Authentication schemes fail when the adversaries pay reconnaissance efforts; an attacker could authenticate all of its bots before the attack. A simple approach to prevent such reconnaissance effort is to frequently re-authenticate users. In general, we do not consider end-host authentication

schemes can defend against fast-path attacks that focus on flooding the network links; an end host has no control over network traffic traveling to itself on the Internet.

4.1.4 Victim-Network Prevention: Resource Accounting and Isolation. As an adversary could pay a reconnaissance effort to authenticate her bots and launch a slow-path attack, to address such an issue, one can choose to understand the server resource allocations and apply fair shares to the clients. Spatscheck *et al.* [58] proposed Escout, an extension to the Scout operating system. Escout allows its users to measure the system resource spent on each client request at different system layers. Using an HTTP request as an example, in Escout, it tracks resource usage, i.e., computation, memory, and I/O, throughout the entire process to complete the request. In other words, the system provides fine-grained telemetry information about resource usage at the different network layers of the server. With such resource telemetry information, Escout allows the system administrator to apply custom Quality-of-Service (QoS) policies to assign fair shares of resources to the clients.

4.1.5 In-Network Prevention: Absorbing DDoS Traffic. Many DPS providers today often use anycast techniques (BGP or DNS) in combination with their content distribution networks (CDN) to increase the overall request processing capacity to absorb DDoS traffic. Depending on the anycast technique in use, such a solution may require DPS providers to hide the protected server IPs from the public, i.e., DNS-anycast-based prevention solutions. Therefore, the adversaries cannot send attack traffic directly to the protected servers. Such a prevention technique is like an arms race between the attacker and the defender; the defender must acquire more resources than the attacker to absorb/fight off the attack traffic from many locations.

Freedman *et al.* [59] proposed an anycast framework for any service. The authors implemented a DNS anycast service and redirect client traffic based on the measured network latency; Each client gets assigned to a server with a low network latency. Gilad *et al.* [60] presented CDN-on-Demand, a system that automatically deploys CDN nodes at cloud providers to absorb DDoS or flash crowd traffic. The system is essentially an open-source implementation of the DDoS prevention solutions offered by DPS providers. The authors of CDN-on-Demand show that it incurs only a few US dollars per month while commercial solutions are considerably higher than CDN-on-Demand. FastRoute [61], as shown in Figure 8, is a BGP-anycast-based CDN system that traps regional network requests to nearby data centers. FastRoute monitors the utilization of each CDN node, and collocates DNS and CDN proxy services at the same location to allow CDN networks to perform fine-grained load balancing scheme at each data center. A FastRoute CDN node announces BGP anycast messages when it is underutilized and withdraw the anycast messages when it is overloaded. Similar to [60, 61], DeDoS [62] is a system that automatically populates the busy modules of an application in the cloud. DeDoS requires software developers to write modular programs in a specific approach, and DeDoS then duplicate the highly utilized modules in the cloud. In case an application-layer DDoS attack happens, DeDoS immediately spawn multiple copies of the program modules that are being affected the most.

In many scenarios, the prevention scheme discussed above is sufficient to protect small websites from common DDoS attacks. Note that for the solutions that employ DNS anycast, they only work when the real IP of the protected servers is hidden from the public. However, recent research projects, e.g., Jin *et al.* [63], allows the adversaries to retrieve the IPs of the protected servers. Thus, the adversaries can bypass CDN prevention schemes and attack the protected servers directly.

4.1.6 In-Network Prevention: Dodge DDoS by Shifting Service Locations. Today, many hosts their Internet services on private or public clouds at fixed locations. However, this property gives the attacker ample time to prepare sophisticated attacks. To avoid such a vulnerability, researchers proposed defense systems [64, 65, 66, 67, 68, 69] to allow service owners to automatically relocate

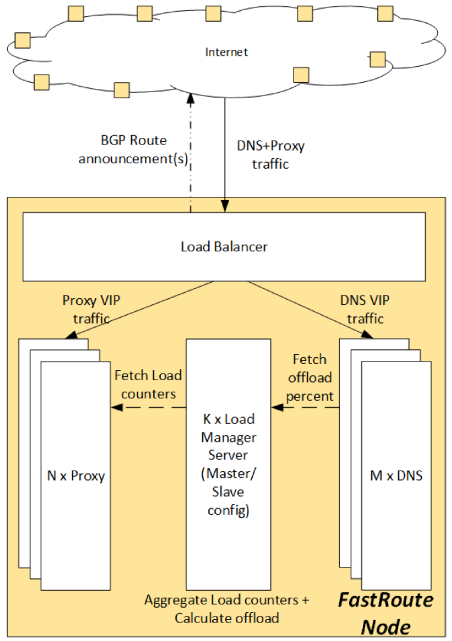

Fig. 8. FastRoute Architecture [61]

their services to different physical/logical locations during the attack time. We call these systems moving target defense (MTD) systems. The goal of an MTD system is to 1) shuffle the clients of a service within a set of service replicas and 2) isolate attack bots onto a small set of replicas while providing legitimate clients stable services with another set of service replicas [70, 71]. While these MTD solutions built on top of overlay networks, the main advantage of MTD over the aforementioned overlay-network schemes is that the MTD solutions do not need to match the resources of the attackers; thus, MTD solutions incur less defense cost.

Jafarian *et al.* [64] proposed OpenFlow Random Host Mutation (OF-RHM), a technique that utilizes the flexibility in SDN to prevent scanners from learning the potential vulnerabilities of a server by frequently shifting the server's virtual IPs. OF-RHM generates virtual IPs randomly to confuse the existing scanning tools, and it provides transparent connection handoffs to the real IPs that are associated with the virtual IPs. MOTAG, proposed by Jia *et al.* [65] in 2013, is a prevention system that defends against link flooding attacks. In MOTAG, clients first authenticate themselves to obtain proxy node IP addresses (exclusively known by the authenticated users). Each client is then provided with one proxy IP, and proof-of-work is applied to protect the client's authentication channel. Whenever a proxy is under attack, it will be replaced by another replica at a different location, and some users will be migrated to the new replica. If the new replica is later under the attacker, MOTAG repeats the same process to move partial users to another replica until there is no attack on the proxy or the insiders (who behave like benign users that leak the proxy locations) are identified. Although MOTAG is a promising prevention system, it requires end hosts to install custom software to solve cryptographic puzzles (PoW) to obtain an access token to the proxies. Thus, MOTAG is not a transparent solution to many, and this is a severe deployment problem that challenges other DDoS defense systems. Authors also have a strong assumption that the attackers can not flood the authentication servers, and victims can reserve numerous servers with low proxy

instantiation time. Similarly, Khattab *et al.* [72] proposed a solution that applies group-testing theory to quickly identify the abnormal clients that utilize the most server resources, and isolate them. Since the MOTAG system does not address slow-path attacks, Jia *et al.* [66] later proposed a system to improve MOTAG. The work [66] introduces a load balancer component that monitors the system loads of overlay nodes and dynamically creates replica nodes to isolate bots onto a small set of machines. [68, 69] are two MTD solutions that focus on evaluating the shuffling mechanism of MTD. They utilize SDN to protect applications that are hosted on cloud providers specifically. However, they both share strong assumptions as above MTD solutions.

4.1.7 In-Network Prevention: Patch and Redesign the Internet. The Internet architecture was not designed with security in mind, and its open design has led to countless DDoS attacks. To address the design issues, network researchers proposed many overlay-network-based systems to address the design flaws in the existing Internet. The main idea of these overlay-network-based solutions is to authenticate legitimate users and only allow authenticated traffic to reach a victim's network. These solutions, by in large, are composed of static overlay networks and can be used for addressing both fast-path and slow-path attacks (Section 3.2).

Yaar *et al.* [73] proposed SIFF in 2004, a system that divides network traffic into privileged and unprivileged classes, where the privileged class has access to higher network bandwidth. SIFF clients have to authenticate themselves with the servers they try to communicate with and obtain tokens that allow them to access the privileged network. In the same year, Keromytis *et al.* [74] proposed secure overlay services (SOS) to prevent DoS attacks proactively. SOS consists of a set of servlets (servers that act as network proxies) that are located around the victim server. Each servlet has an "approved" IP address to access the protected network; SOS filters all traffic without a valid source IP address. Clients in this system have to authenticate themselves with one of these servlets; only then, these clients can forward their requests to the protected network. Because SOS filters unauthenticated traffic before they reach the actual victim server, link flooding attacks are less likely to incur in the SOS network. SOS assumes the list of "approved" IPs will stay secret, and the attackers cannot spoof the IPs to reach the protected network. Although SOS shows promising prevention results against link flooding attacks, it requires end-host software modifications to route their traffic via the servlets.

Andersen [75] later proposed Mayday, a system to improve SOS by separating SOS's overlay routing and filtering. The author argues that SOS [74] is susceptible to insider attacks as an attacker could authenticate her bots in SOS to learn all the *approved* IPs, and the bots can spoof their addresses as the *approved* IPs to reach the real victim server. Mayday, on the other hand, supports a set of overlay routing schemes to prevent attackers from learning the *approved* IPs. For example, Mayday can use either 1) random routing or 2) doubly-indirect routing, which adds another routing layer in the overlay network to mitigate the insider attacks. However, these additional protection mechanisms add both computational and network overhead on the overlay nodes (or servlets in SOS). In delay-sensitive services, e.g., Voice over IP (VoIP), Mayday will not be their best DDoS prevention choice. Around the same time, WebSOS [76] was proposed as a prevention system that integrates CAPTCHA [55] and SOS [74] to protect web servers from the attacks while requires little modifications to the end-to-end web software stack, e.g., web browser and server.

Stavrou *et al.* [77] proposed a DDoS attack, the sweeping attack, against the overlay-based prevention solutions and provided a stateless overlay network to address the problem. In the sweeping attack, the attacker uses her bots to attack a small set of proxy nodes in an overlay network at a time. Thus, the targeted nodes are affected by fast-path attacks. The attacker frequently changes the set of nodes to attack so that the latency-dependent services will be severely affected. E.g., imagine VoIP calls get interrupted every 10 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 the system to send their packets across multiple proxy nodes in the overlay network. Overlay nodes then forward the packets to the protected server (this technique shares a similar idea behind Multipath TCP [78]). Unlike Multipath TCP, *spread-spectrum* allows clients to send *duplicated* packets across multiple overlay nodes to ensure a higher packet delivery rate in case the DDoS traffic congests some inbound links of the overlay nodes. This approach works when the attacker cannot overwhelm *all* the overlay nodes simultaneously; this assumption is close to reality as it is challenging for a botnet to overwhelm physically distributed nodes on the Internet. The authors evaluated the performance of the work on PlanetLab testbed. As a result, the throughput reduction rate ranges from 50% (no packet replication spreading) to close 10% (when each client replicates each packet twice). In summary, *spread-spectrum* is an interesting technique to deliver traffic through in overlay networks. However, this technique may cause cascading failure to the Internet when it is widely used, i.e., the effective bandwidth of each link shrinks as the packet replication rate goes up.

Dixon *et al.* [79] proposed Phalanx to defend against slow-path attacks. Authors recognized that DDoS classification is a difficult task as many DDoS attacks today behave just like benign clients. Authors argue that while the previous defense solutions, some while elegant, all have high deployment burden. Phalanx itself is not much deployable as it requires software support from end to end, meaning that it requires custom client, server, router, and overlay software. In Phalanx, when a client sends packets to a destination, its packets traverse through a set of random traffic forwarders/nodes, and the attacker is unlikely to know the correct forwarding path. The system then only delivers the packets forwarded by the forwarders to the destination, so only the legitimate requests can go through. It leverages a set of machines that absorb DDoS traffic, and it assumes the aggregate capacity of the machines does not exceed the botnet by a huge margin. The system is not always on, and it stops when the attacks become ineffective. Similar to SOS and Mayday, Phalanx network drops unauthenticated packets by default, and there are two ways to authenticate a client: authenticated tokens exchanged off-band and crypto-puzzles (proof-of-work). In particular, a client can directly access any Phalanx-enabled server if the client has an authenticated token of the server. If the client does not have an authentication token, the client can still access the server by solving crypto-puzzles (or proof-of-work).

Another overlay-based prevention system, ScoreForCore, proposed by Kalkan *et al.* [80], is a collaborative system that proactively removes DDoS traffic when network congestion is detected. Each router in the overlay network determines whether a packet is legitimate through Bayesian Theorem; if the value of a packet exceeds a predetermined threshold, the packet is then considered as a part of the DDoS attack. While ScoreForCore is similar to Mayday [75], in that it forms a set of protection rings around protected destination networks, ScoreForCore does not handle DDoS that employs IP spoofing as the routers have to maintain flow states correspond to each packet. In this case, the ScoreForCore system runs out of the system memory to maintain scores for all flows.

MiddlePolice [14] includes a prevention scheme that redirects the traffic towards a destination. MiddlePolice uses BGP or DNS anycast to enforce the traffic forwarding path towards a destination. All directed traffic goes through a set of *mboxes*, a set of commodity servers that police traffic. The *mboxes* then process the traffic towards the destination and maintain the statistics of each traffic flow. The destination network chooses a bandwidth allocation policy to ensure its service quality; e.g., the network could enforce an equal amount of bandwidth to each sender or even allocate bandwidth at AS-level granularity. In MiddlePolice, the real IP of the destination network is hidden from the public. In case the real IP is leaked, MiddlePolice requires the network provider of the destination network to install ACL rules to forward only the packets processed by mboxes

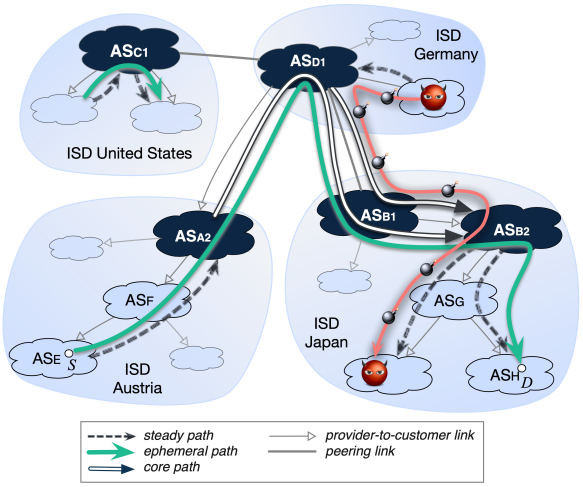

Fig. 9. Sibra [82] Architecture

to the destination network. For mboxes to work effectively, each mbox must learn the available bandwidth from itself to the destination network. MiddlePolice builds a feedback loop between each mbox and the destination network, and the destination network can report each mbox the available bandwidth individually. As mboxes are general-purpose machines, the packet processing performance is inefficient when compared to the hardware routers, especially when each machine needs to track flow states while rate limit flows.

STRIDE [81] and Sibra [82] are two similar DDoS prevention schemes that defend against fast-path attacks. Both solutions rely on SCION [83], which stands for *Scalability, Control, and Isolation On Next-Generation Networks*, to perform long-term bandwidth allocation among transit networks, and both systems ensure short-term end-to-end link bandwidth between two hosts. Due to the end-to-end bandwidth allocation, the network availability is *independent* of the size of botnets. Figure 9 shows Sibra [82] system in action. SCION is a clean-slate Internet architecture that considers many missing features of the Internet today. E.g., in SCION, it includes mechanisms to prevent IP spoofing and route hijacking from happening. However, at the time of this writing, the deployment of SCION is at its initial stage, and whether the Internet will adopt it is too early to tell.

4.1.8 DDoS Prevention Summary. In Table 3, we qualitatively rank each selected prevention solution's performance, scalability, and deployability. As the first and the most critical defense stage, we realized there are many methods to provide a more secure network environment. While it is impossible to design systems with perfect security, we need to reduce the size of attack vectors to prevent large-scale attacks from happening. We should not only rely on device manufacturers (end-host security) to produce devices with better security and upgradable, but different networks also need to implement the best preventative practices.

Most solutions in Sec. 4.1.5 (absorbing DDoS traffic) and Sec. 4.1.7 (patching and redesigning the Internet) deliver good to excellent performance in DDoS prevention. Such a solution traps attack traffic in regional networks before the attack traffic accumulates at the victim-end networks. While these solutions hardly run into any scalability issue to a small number of users/networks, the solutions can impose huge heavy workloads on the clouds, routers when facing more users/networks. E.g., cloud providers may not be able to absorb traffic for 1000s of edge networks simultaneously,

Table 3. Selected DDoS prevention solutions

	Main Idea	Example	Performance	Scalability	Deployability
Source-End (Sec. 4.1.1, Sec. 4.1.2)	End-Host Security	Out of scope	N/A	N/A	N/A
	IP-Spoofing Prevention	Ingress filtering [33]	Excellent	High	Medium
		Park *et al.* [53] Mirkovic *et al.* [54]	Fair	Medium	Low
Victim-End (Sec. 4.1.3, Sec. 4.1.4)	Authentication	CAPTCHA [55]	Good	High	High
	Resource Accounting	Escout [58]	Good	High	Medium
In-Network (Sec. 4.1.5, Sec. 4.1.6, Sec. 4.1.7)	Absorb Attack	OASIS [59] CDN-on-Demand [60]	Good	Medium	High
		FastRoute [61]	Excellent	Medium	High
	Dodge Attack	MOTAG [65] Khattab *et al.* [72]	Fair	Medium	Medium
		OF-RHM [64] Jia *et al.* [66] Debroy *et al.* [68] Shan *et al.* [69]	Fair	Low	Medium
	Patch/Redesign Internet	Phalanx [79] ScoreForCore [80]	Fair	Medium	Low
		WebSOS [76] Mayday[75] SOS[74]	Good	Medium	Low
		Stavrou *et al.* [77]	Good	Low	Med
		STRIDE [81] SIBRA [82]	Good	High	Low
		MiddlePolice [14]	Excellent	Medium	Medium

and the use of careless BGP anycast could impose an increased network footprint on the Internet. Indeed, Morau *et al.* [6] presented a preliminary study of the collateral damages caused by BGP-anycast-based defense solutions. Although these solutions can be costly, we value their deployability from medium to high as they use the most practical technique that one can deploy on today's Internet. The redesigning-the-Internet solutions [81, 82], on the other hand, face challenges such as deployment incentives and cost. Thus, we value their deployability as low. With that said, they are reasonable prevention solutions in terms of their performance, and they are inherently scalable due to their design.

Moving target defense (MTD) solutions (dodging attacks), utilize overlay networks, deserve a new category in DDoS prevention due to their unique operational model. These solutions allow users to combat DDoS attacks with a finite amount of resources and regionalize the attack power via group testing. However, all reviewed MTD solutions have assumptions that may be difficult to implement in the real world. For example, the lack of considering application state migration procedure, and the existing MTD solutions assume a fixed number of insiders that can inform the adversaries as to render the defense techniques ineffective. For these unaddressed reasons, we value their performance as fair, and deployability as medium. We rank their scalability as medium as such a solution requires each overlay node to maintain utilization states of each client.

While researchers have developed many DDoS prevention solutions to protect networks from different aspects, as we listed above, these prevention solutions are certainly not comprehensive enough or cost-effective [60, 61] for defending against the latest attacks. We need to detect/classify the latest or even unknown attacks and mitigate them.

4.2 DDoS Detection and Classification

While prevention schemes are crucial to DDoS defense, they cannot protect all types of servers/networks. When they fail, the victims first need to detect/classify the DDoS attacks before mitigation. In this section, we review each selected detection/classification solution using the same template as defined in the section overview. As we mentioned in Sec. 2.4, DDoS detection/classification techniques are agnostic to deployment locations. The main challenge is to provide them with a sufficient amount of telemetry data. We summarized all the reviewed solutions in Table 4.

4.2.1 Identify Anomalies/Attacks. Before mitigating any attacks, we need to know whether there is an attack. We review a list of papers that focus on detecting the attacks. While these solutions do not facilitate fine-grained DDoS mitigation, they serve as a preliminary step towards DDoS mitigation nonetheless. *In our study, we only find detection solutions that are designed for* ***victim networks****, in other words, they assume a complete view of telemetry data that is available to them.* The selected DDoS detection solutions are as follows:

General network anomaly detection solutions, such as [84, 85, 86, 87], derive the threshold values of the monitored networks when the networks are healthy, and they model DDoS attacks as simple as *a large number of connections towards one destination*. However, as discussed in Sec. 3, the latest attacks send "legitimate" traffic to the victims, and some attack traffic may never reach a victim's network [17, 18]. These general anomaly detection systems are not applicable in these cases.

Hussain *et al.* [88] proposed a framework to detect link-flooding attacks in victim networks. The work also determines whether an attack comes from a single host or multiple hosts. The authors first apply packet header analysis that uses packet ID and TTL fields to infer whether packets are spoofed, thus to detect attacks. However, the adversaries can easily mimic benign packets to render the technique ineffective. The authors then analyze the temporal relationships of the traffic volume to monitor the traffic ramp-up behavior: if traffic ramp-up behavior exists, the authors believe it provides a good indication that there is a multi-source attack. As smart attackers can create artificial ramp-ups, the authors proposed a spectral analysis technique to improve the robustness of detection. The analysis counts the number of attack packets in 1ms intervals over a time window, e.g., 30 seconds. For each time window, authors compute power spectral density using the known attack traces, i.e., single- and multi-source attack traces. According to their evaluation, single- and multi-source attacks behave rather differently. Hussain *et al.* [89] later presented an automation process to fingerprint and identified repeated DDoS attacks. The process relies on the spectral analysis technique from their previous work [88]. COSSACK [90] is a source-end defense solution that relies on 1) traceback solutions to identify attack source networks, 2) Snort, to detect attacks based on predefined thresholds, and 3) detection techniques from [88] to identify attack types. In COSSACK, the victim networks provide attack-source networks the attack detection criteria, and the attack-source networks then monitor for the potential attacks. The COSSACK system scales as it distributes the detection load across all edge networks. Similar to many source-end defense solutions, this work does address deployment issues. Furthermore, [88] may not work well under the operation context of COSSACK, as link-flooding attacks generally do not occur at attack source networks.

Gavrilis *et al.* [91] analyzed DDoS attack tools and found many of the tools spoof *source IPs*, *source and destination ports*, and other *flag fields*. With the discovery, the authors proposed an attack detection system that relies on statistical features, e.g., TCP sequence number, window size, to train radial-basis-function neural networks in the victim network. This system, however, only detects the common attacks as covered in Sec. 2.2. To understand what packet features in a network are more important for DDoS detection, Osanaiye *et al.* [92] proposed an ensemble-based feature selection method to choose the best feature set of a network for attack detection. [92] 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 for the network. Unfortunately, the work does not detail what DDoS attacks it detects.

Rather than relying on traffic volume analysis [88, 89, 90] to detect attacks, many entropy-based approaches [93, 94, 28, 95] use packet features, e.g., IP addresses, ports, and flow volume to build detection systems with more insights. Nychis *et al.* [93] 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.* [94] proposed an entropy-based DDoS detection system for web servers. The goal is to detect attacks that utilize legitimate HTTP requests. The system calculates the popularity of each accessed documents 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 evaluate whether the trained system can later detect those injected attack traces. This work is only applicable to websites that host static web content, and it does not detect non-HTTP-based attacks. Xiang *et al.* [28] proposed an entropy-based detection method to detect low-rate DDoS attacks, with strong assumptions such as the full control of network routers on the Internet, and the attack traffic follows the Poisson distribution. [95] is another entropy-based detection system, and the authors evaluated their solution against 3 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.

4.2.2 Identify Suspicious/Attack Flows. Only knowing a DDoS attack is occurring is not enough, we must classify attack flows to facilitate a fine-grained DDoS mitigation. In theory, we can define a flow in a traditional sense: n-tuple packet header fields; one can also define a flow by the shared signature/pattern among the attack traffic. In reality, an adversary can render the latter infeasible by generating packets with random/encrypted content. For example, the adversary can launch a CrossFire-like attack that sends seemingly-benign HTTPS traffic.

4.2.2.1 DDoS-Source Classification. D-Ward [11] is a source-end DDoS defense solution that detects and mitigates fast-path attacks. Each router in D-Ward tracks the statistics of each flow (e.g., a TCP flow), and compares the statistics against a benign flow's traffic model. For example, if a TCP flow's packet in-and-out rate lies outside a predetermined threshold, this TCP flow is flagged as attack flow. While the system can detect discernible attack flows at attack traffic source, it cannot detect attack flows that follow benign flow behaviors; due to such behavior, these attacks render source-end attack detection ineffective. Lu *et al.* [96] proposed a source-end detection system that uses both threshold-based and machine learning-based approaches to detect SYN attacks at edge networks;

4.2.2.2 Victim-Network Classification. Sun *et al.* [37] proposed a distributed system to detect slow-path pulsing attacks at/close to the victim end. The system requires all the upstream routers of an attack victim to perform detection on their egress ports toward the victim. To detect the pulsing attacks, authors frequently sample a network's traffic, e.g., 100 times per second, and collect the time series of the network's link bandwidth utilization. The system then scans for the bandwidth utilization windows that are bursty, and verify whether there is a pattern of the bursty windows. Because the system requires transit AS deployment, the cost of the detection is not trivial;

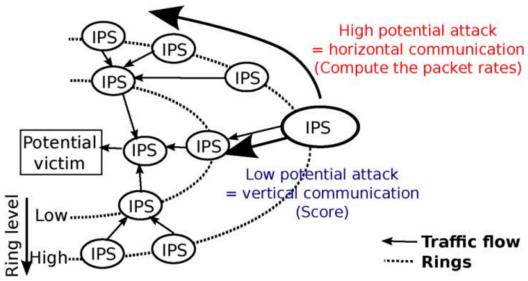

Fig. 10. Firecol [102] protection rings

the computational complexity of the detection algorithm is $\theta(n^2)$, and a transit network needs to provide always-on detection for all its customers.

DDoS Shield [97, 98] is a defense solution that aims to address slow-path attacks, and authors used HTTP servers as their proof of concept. DDoS Shield tracks individual HTTP client's behavior with metrics such as request inter-arrival time, request workload on the DDoS Shield-enabled server. DDoS Shield then calculates a suspicion value of each HTTP client based on the behavior metrics and triggers the mitigation process if necessary.

Khattab *et al.* [72] proposed a defense system that uses group-testing theory to bait/detect and mitigate fast-path attack bots. The work applies threshold-based detection and assumes attack traffic is significantly different from benign traffic. The work is applicable to detect small-scale DDoS attacks with a limited amount of bots; each bot has to send more requests to overwhelm server resources.

Afek *et al.* [99] proposed an attack signature extraction method for link-flooding attacks. The goal of the work is to find a set of strings (attack signatures), S, that only exists in a network when it is under link-flooding attacks when compared to the network's normal operation. However, this work only works when the network traffic is unencrypted, which means it is not suited for many DDoS attacks that employ spoofing or encrypted attack flows.

4.2.2.3 In-Network Classification. LADS [100] is an in-network DDoS defense system that detects fast-path attacks. The system relies on traffic measurement tools that are available to most ISPs: SNMP and NetFlow. LADS consists of a lightweight and heavyweight detection stages. In lightweight detection, it monitors router traffic volume, CPU utilization, and packet drops. The heavyweight mode is triggered when the lightweight detection process finds anomalies. In the heavyweight mode, the system categories traffic into TCP SYN flows, ICMP flows, ICMP flows, etc. The system then checks the flow statistics against a set of predefined thresholds to alarm the network operators with affected destination networks. This two-stage detection system also implies that an attack will not be detected if the lightweight detection stage fails to catch the attack.

Zhang *et al.* [101] proposed an in-network detection system to classify attack flows in pulsing attacks, and the detection system requires a high deployment rate at transit networks. The system exploits the fact that the legitimate flows in a pulsing attack tend to reduce their traffic sending rate during traffic congestion periods. Based on this observation, the system applies packet sampling to monitor the bandwidth utilization of all 5-tuple flows (src./dest. IPs, src./dest. ports, and protocol) in a network. The system looks for the flows that repeatedly appear in multiple congestion periods that do not reduce their traffic sending rates, and classifies them as attack flows.

Francois *et al.* [102] proposed Firecol. A system deploys intrusion prevention systems at the ISPs to form overlay networks that protect the subscribed customers. The overlay networks form

protection rings around the customers, as shown in Fig. 10, and each node in a ring can talk to any other nodes in the ring to exchange traffic information and to better detect DDoS attacks. To this end, the authors of Firecol argue that the attack victims do not have to deploy an IPS that can process high volume attack traffic. Layers of the protection rings surround the attack victim in Firecol. Firecol employs a score-based attack traffic detection method. The system keeps track of the 1) frequency of each network flow, e.g., 5-tuple flow, as well as 2) the frequency distribution of all the flows. Authors believe a network with high-frequency flows and skewed frequency distribution is a good indication of an attack. However, as the system maintains states for every flow for all subscribed customers, the adversaries can generate an overwhelming amount of flows to overrun the detection system of Firecol.

Xu *et al.* [103] and Zheng *et al.* [104] explore the viability of using SDN to measure traffic statistics and detect slow-path attacks. In particular, Both solutions utilize the limited OpenFlow rule space to perform threshold-based attack detection without traffic monitoring tools such as sFlow and NetFlow. However, both [103, 104] provide little motivation for using OpenFlow to perform attack detection when the network community has mature telemetry solutions such as sFlow and NetFlow. Xu *et al.* [103] detect link-flooding attacks by a threshold-based anomaly detection scheme. It first monitors flows with abnormal flow bandwidth or packet rate, and each flow is described by a large network prefix, e.g., /16. Rather than using SDN for DDoS defense, SPHINX [105] analyzes the vulnerabilities with existing SDN frameworks and provided threshold-based solutions to detect slow-path attacks in SDN networks, e.g., switch TCAM exhaustion, DoS on SDN controllers, etc.

Authors of SPIFFY [106] recognized that DDoS attack flows could be indistinguishable from legitimate flows, and argued that defense should consider the following techniques: 1) force attackers to increase their costs, 2) withstand the attack, 3) force attack traffic to become distinguishable from legitimate traffic. SPIFFY requires a transit network router to temporarily increase a victim's link bandwidth and observes the behaviors of the traffic flows toward the victim. Authors assume that attackers set a maximum traffic sending rate of each bot to save cost, and after the victim's link bandwidth is increased, the attack flows not increase as the what a benign flow should.

4.2.3 Section Summary. We summarized DDoS detection and classification solutions in Table. 4. While general network anomaly detection solutions [84, 85, 86, 87] are extremely efficient, they require network operators to tune the thresholds for different networks to work carefully. They are not suited for detecting more advanced attacks. Machine-learning-based detection systems [91, 92], in theory, could adapt to different network environments, the two solutions reviewed are built on top of over-simplified DDoS models. Entropy-based solutions [93, 94, 28, 95] use more traffic metrics to profile a monitored network. We believe entropy-based solutions offer the best detection results out of all reviewed detection solutions in this report. While the detection process in COSSACK [90] is scalable and can offer better insight into the attack traffic, the deployability of COSSACK is low as it requires deployment at DDoS-source networks. For the classification solutions, we find that most temporal-analysis-based solutions [37, 97, 98, 101] have good detection results for the attack flows that they intended to classify. The reviewed classification solutions are also scalable in their operational contexts. We rank the performance of Afek *et al.* [99] as low since it assumes attack traffic consists of plain text that comes with discernible patterns.

We found that DDoS detection/classification remains a challenge as adversaries continue to discover new attacks; the network community lacks sound detection/classification solutions for the latest attacks as we reviewed in Sec. 3. Many such systems have over-simplified models or unrealistic assumptions of the DDoS attacks. For example, many assume the attack concentrates all

Table 4. Selected DDoS detection and classification solutions

	Main Idea	Example	Performance	Scalability	Deployability
Detection					
DDoS-Source	N/A	None	N/A	N/A	N/A
Victim-Network (Sec. 4.2.1)	Static thresholds	Lakhina *et al.* [84] Soule *et al.* [85] Li *et al.* [86] Silveira *et al.* [87]	Poor	High	Medium
	Machine learning	Gavrilis *et al.* [91] Osanaiye *et al.* [92]	Fair	Medium	High
	Temporal analysis	Hussain *et al.* [88, 89]	Good	Medium	Medium
		COSSACK [90]	Good	High	Low
	Traffic entropy	Nychis *et al.* [93] Xie *et al.* [94] Xiang *et al.* [28] Behal *et al.* [95]	Good	Medium	High
In-Network	N/A	None	N/A	N/A	N/A
Classification					
DDoS-Source (Sec. 4.2.2.1)	TCP profiling	D-WARD [11]	Good	High	Low
Victim-Network (Sec. 4.2.2.2)	Str. pattern recog.	Afek *et al.* [99]	Poor	Medium	Medium
	Baiting	Khattab *et al.* [72]	Fair	High	High
	Temporal analysis	Sun *et al.* [37]	Good	Medium	Medium
		DDoS Shield [97, 98]	Good	Medium	Low
In-Network (Sec. 4.2.2.3)	Static thresholds	LADS [100]	Fair	High	High
	TCP profiling	Zhang *et al.* [101]	Good	Medium	High
	Temporal analysis	FireCol *et al.* [102]	Good	Medium	High
	Baiting	SPIFFY [106]	Good	High	Medium

attack power towards one host/IP. They use static usage threshold values obtained at training time, and the attempt to extract attack signatures from packet payload while most traffic is encrypted.

At the same time, research projects often overlook their deployment issues. Depends on the operational context, many projects face a *base rate fallacy* problem: claiming a system has low false positives is a preliminary step towards a good detection system. In real networks, the network operators cannot blindly trust a detection/classification system to automate the DDoS mitigation. They need to examine generated alerts to confirm the attacks. Unfortunately, none of the detection papers we reviewed remotely considered this aspect.

The detection/classification solutions discussed above do not focus on the runtime speed in the real world, and they are not optimized to work with high-bandwidth links. Although we have mature telemetry solutions such as NetFlow and sFlow, they are not optimized for fine-grained DDoS detection tasks. Yoon *et al.* [107] and Yu *et al.* [108] proposed systems to utilize router SRAM and TCAM resources to monitor attacks more efficiently, e.g., OpenSketch [108] utilizes FPGA board to track the number of unique connections per host at high-speed network links. These solutions complement the above detection/classification solutions.

Overall, we believe an ideal detection/classification solution should incorporate better attack models, carefully address the base rate fallacy issue as mentioned above, and define detection/classification results clearly as they directly facilitate the mitigation process.

4.3 DDoS Mitigation

While DDoS victims can mitigate many slow-path attacks locally, as these attacks do not deplete the link bandwidth, the victims must seek mitigation help from their upstream networks to handle

the attacks otherwise. Because each network on the Internet operates autonomously, it is not trivial to design a comprehensive mitigation system that works with different network environments. For this reason, in this section, we only review a list of prominent **DDoS-source** and **in-network** mitigation solutions. We cover the primary design of each mitigation solution, and we further divide in-network mitigation solutions based on the required level of system deployment rate as follows:

- *In-line* systems that require transit ASes to participate
- *Redirection-based* systems that require a low system deployment rate

4.3.1 DDoS-Source Mitigation. For any DDoS-source mitigation system to work effectively, it requires a much higher deployment rate in-network solutions. Some even require end-hosts to participate [109]. On the other hand, mitigation solutions under this category are inherently more scalable than others. A DDoS-source solution must offer reasonable incentives for the edge networks or end hosts to deploy. We explore the highly-cited solutions in this section.

D-WARD [11] requires system deployment at traffic source routers, e.g., a home router. This deployment location enables D-WARD to track a few flows without imposing significant network performance penalties. D-WARD imposes rate limits on these attack flows and continuously monitors their behaviors. Once the traffic profiles of these flows fit within the range of benign traffic profiles, D-WARD lifts the rate limit on the flows or the host.

COSSACK [90] requires edge network deployment, and each attack source network then collaborates with the DDoS victims to mitigate attacks. The goal of COSSACK is to determine the responsible networks of an attack than individual bots, and it relies on IP traceback systems to work effectively. A similar approach to COSSACK, Huici *et al.* [110] proposed a mitigation system that requires edge networks to establish IP-to-IP tunnels with the edge network that hosts services and reject all traffic that is not coming from the tunnels. Once the tunnels are established, the victim edge networks can know where the attack traffic, spoofed or not, are coming from, thus allows the victims to ask the traffic source networks to filter unwanted traffic.

Speak-up [109] takes a unique approach in mitigating DDoS attacks by having the clients of a victim server send a higher number of requests during attack time — a bandwidth competition between benign users and attackers. The goal of Speak-up is to allow benign clients to acquire fair shares of the link bandwidth towards the victim server. The system requires software modification on end hosts and servers, and in the case when a server is oversubscribed, Speak-up *encourages* its clients to send redundant requests to "crowd out" the attacking requests. One big assumption in Speak-up's threat model is that the attackers always use up most of their attack capabilities, and cannot cope with the *encouragement* from Speak-up servers.

CoDef [111] is a collaborative defense system with two main techniques: 1) collaborative traffic rerouting via existing inter/intra-domain routing protocols, and 2) collaborative rate control at both flow level and AS level. In collaborative rerouting, CoDef allows the victim AS to send traffic rerouting requests to traffic-source ASes, and redirect the traffic towards the victim to rate controlling nodes. In collaborate control, CoDef rate limits bot-contaminated ASes at AS level, and commands uncontaminated ASes to prioritize their flows and rate limit the flows based on their priorities. CoDef claims the system is practical to deploy in today's Internet. However, like many other DDoS defense systems, CoDef does not provide strong incentives for ASes to deploy such a solution.

4.3.2 In-Network Mitigation: In-Line. The performance of a solution in this category highly depends on the transit network deployment rate. Such a solution does not proactively interrupt the default

routing behavior on the Internet. The traffic policing/filtering load at each AS reduces as the deployment rate of such a mitigation system increases.

Jin *et al.* [112] recognized that attackers could spoof any byte of a packet. However, it is more difficult to spoof the time-to-live (TTL) field, i.e., spoofed packets are likely to traverse a different number of hops than from the legitimate networks. Thus, the authors proposed a scheme to infer the TTL values of the packets from genuine networks, and the system only accepts packets from a source network with the expected TTL value(s). However, this mitigation system does not provide a guarantee on the false positive/negative rates, e.g., the system does not handle situations such as route changes. Also, it is highly likely for an attacker to figure out the correct TTL values for the packets he/she needs to spoof; DDoS bots in a massive botnet are often geographically distributed.

AITF [12, 113] utilizes the filtering capabilities in transit routers, e.g., AITF can use hardware ACL to matching network flows and remove them. By default, AITF filters attack traffic at the routers that are closer to the traffic sources. The authors recognize the TCAM space in network routers is limited, and suggest that if each AITF-enabled router can allocate a few thousand hardware filters for traffic filtering, the whole system can support fine-grained DDoS filtering with millions of filters. The AITF-enabled routers implement an IP traceback system that stamps their IP addresses to each packet. Thus, the mitigation systems can place DDoS filters at the responsible routers. AITF justifies its deployment incentive for ISPs as follows: an ISP can either participate in the AITF system or lose its connectivity to the victim during a large-scale DDoS attack where the victim can employ RTBH for mitigation. Similar to AITF, StopIt [114] is another in-line mitigation system. StopIt assumes good detection solutions exist, e.g., via CAPTCHA-like systems. It 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 mitigation effort. To avoid such an attack, StopIt relies on the authors' previous work, Passport [115], to prevent IP spoofing. Unfortunately, Passport [115] does not provide a strong incentive for network communities to deploy.

Ballani *et al.* [116] 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 announced its reachability. Because each router in this system needs to maintain the announcements from end-hosts, these routers may not have enough low-latency memory (CAM/TCAM) to maintain network throughput.

DefCOM [13] is an overlay-based defense system that includes both attack detection/classification and mitigation. The attack detection/classification component is deployed at victim networks while the mitigation component is deployed in the transit networks. DefCOM requires the transit routers to deploy its mitigation component and impose rate limits on the attack flows, 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.

While most in-network defense systems focus on inter-AS collaboration, dFence [117] is an intra-AS DDoS defense system to enable ISPs to provide DDoS mitigation services to their customers within its capability. Authors in dFence realized that many proposed DDoS systems require either software modification at either router or end host or both. Thus, dFence aims not to change anything in the existing network infrastructure; it directs the victim's traffic to middleboxes via intra-domain routing algorithms. The middleboxes in dFence offer both stateless and stateful traffic filtering. Similar to [118, 14], dFence employs tunneling techniques to ensure that the bi-directional flows flow through the middleboxes, thus ensures the viability of stateful traffic filtering.

SENSS [119] is an SDN-based defense system that is similar to Bohatei [15] and MiddlePolice [14]. Unlike [15, 14], SENSS is an in-line mitigation solution that does not employ any traffic redirection techniques. SENSS requires each participating AS to implement a set of APIs that provide a DDoS victim to 1) query traffic statistics in its upstream networks and 2) control whether a network

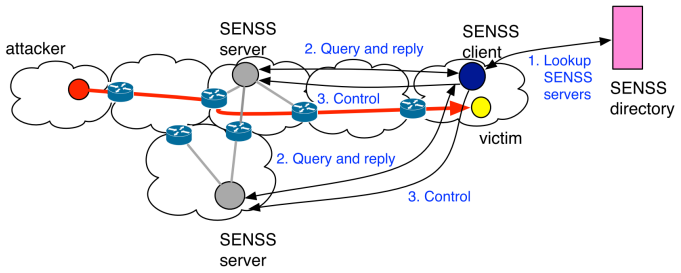

Fig. 11. SENSS Architecture [119]

flow towards the victim should be forwarded. Fig. 11 illustrates the architecture of SENSS. For example, if a victim suspects itself is under a CrossFire attack [18], it first queries SENSS-enabled upstream networks and verifies whether there is any congestion in transit links. It then uses traffic measurement data to nail down the flows that are participating in the attack. Similarly, Steinberger *et al.* [120] proposed a protocol for DDoS victims and ISPs to exchange attack information to avoid the ad hoc approach taken by many ISPs today.

To avoid the deployment issues, Dietzel *et al.* proposed Stellar [16], a mitigation system that leverages existing RTBH infrastructure to disseminate fine-grained filters to major Internet Exchange Points (IXPs); the fine-grained filters are essentially encapsulated in the BGP announcements. As more stub networks join IXPs, authors believe IXPs have a unique advantage in mitigating DDoS attacks on today's Internet.

4.3.3 In-Network Mitigation: Redirection. The mitigation systems in this category rely on 1) anycast techniques to "hijack" (redirect) the victim's traffic, e.g., DNS or BGP, to locations equipped with more resources than the victims, at the *inter-AS level*, and 2) SDN and NFV to distribute and filter the victim's traffic at the *intra-AS level*. While widely adopted by the DPS providers, such a mitigation technique requires additional hardware equipment to mitigate the attacks and may impose negative cascading effects on the Internet [6]. E.g., a transit network's link utilization may experience a sharp increase when many DPS providers use such a technique simultaneously.

CenterTrack [118] is an early work that exploits BGP to redirect the victim network's traffic to an overlay network that is designed to mitigate attacks. The overlay network in CenterTrack tracks the traffic forwarding path of each packet en route to the victim network. Therefore, CenterTrack has the flexibility to choose which router to filter traffic. To ensure that the overlay network can process bi-directional traffic from and to the victim's network, CenterTrack requires the victim network to 1) connect a physical link to the overlay network or 2) establish virtual tunnel, e.g., *generic routing encapsulation* (GRE), with the overlay network to funnel all its traffic through. Similar to CenterTrack, MiddlePolice [14] consists of a mitigation component that redirects attack traffic to data centers via both DNS and BGP anycast. Once the victim's traffic arrives in a data center, each MiddlePolice node in the data center mitigates the attack by using techniques such as rate limiting and packet filtering, as shown in Fig. 12. For example, MiddlePolice implements a per-sender-level bandwidth allocation scheme similar to STRIDE [81] and Sibra *et al.* [82]. If the attacker finds a way to escape the traffic redirection, i.e., the attack traffic does not flow via MiddlePolice data centers, MiddlePolice relies on the transit networks, e.g., the ISPs as shown in Fig. 12, to drop all packets toward the victim network but the packets that have been processed by the MiddlePolice nodes.

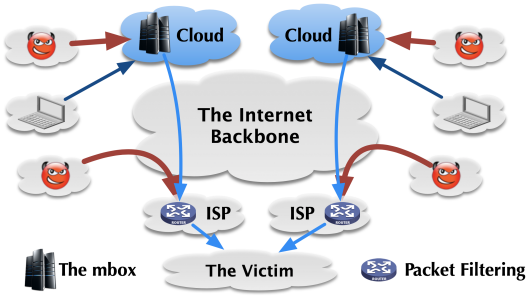

Fig. 12. MiddlePolice Architecture [14]

Bohatei [15] is an intra-AS DDoS defense system that leverages SDN and NFV; each Bohatei-enabled ISP offers DDoS defense as a service to its customers, and when an attack happens, the victim's traffic will be rerouted to data centers for traffic mitigation. During the mitigation time, an ISP first measures the attack volume towards the victim to decide the number of virtual machines (VM) required at different data center locations. The ISP uses MPLS or IP tunneling to redirect traffic to different data centers, where each data center spawns mitigation processes on the allocated VMs and dynamically balances attack traffic load to the VMs.

As many criticize the impracticality of many DDoS mitigation solutions, Smith *et al.* [121] proposed *routing around congestion (RAC)*, a solution to mitigates link-flooding attacks against transit networks, e.g., the Coremelt attack [17], that only rely on carefully crafted BGP announcements. In RAC, a victim network announces BGP poisoning messages to its targeted upstream ASes and hopes to force the selected ASes to divert their traffic to a healthy link. RAC assumes that the victim knows 1) attacks such as [17] is occurring, 2) link congestion locations in transit networks, and 3) there are alternative links to share the attack load. However, later work from Tran *et al.* [122] studied RAC [121], and concluded that RAC is unattainable in the real world.

4.3.4 Section Summary. We summarized the reviewed mitigation solutions in Table 5. We rank the performance of most mitigation solutions as good to excellent since they meet their mitigation goals as claimed. We assign a poor or fair performance score to a mitigation solution when its primary technique relies on strong assumptions. E.g., Jin *et al.* [112] filter traffic based on inferred packet TTL values, while such a technique could negatively affect the normal operations of a network. Similar to the prevention solutions in Sec. 4.1.5, redirection-based mitigation solutions face similar scalability challenges; They are more difficult to scale to protect multiple networks simultaneously. We rank the deployability of the redirection-based solutions to be medium since the system cost is higher for each network who wishes to deploy such a solution. In-line and DDoS-source mitigation solutions are inherently more scalable than redirection-based solutions, as discussed in Sec. 2.4. For this reason, we value the scalability of most such solutions to be medium to high. Finally, we value the majority of the mitigation solutions to have a medium level of deployability. While these solutions are technically viable to deploy, they hardly meet all the metrics for a deployable solution, as defined in the section overview.

We found a majority of the systems do filter or rate limit attack traffic at a fine-grained flow level. More recent solutions such as [15, 14] proposed to filter traffic in software with arbitrary matching patterns. However, none of them take further steps to study the filtering capacity of the Internet as a whole, not to mention finding the optimal level of DDoS filtering granularity in a mitigation system with a limited filtering capacity. We also believe the research community

does not lack mitigation systems, but ways to incentivize ISPs to deploy a mitigation system that supports fine-grained DDoS filtering. We summarized the reviewed mitigation solutions in Table. 5.

Apart from the mitigation systems above, we found solutions from Soldo *et al.* [123, 124] study how to generate efficient DDoS filters with limited filter space, i.e., how to use a single source-prefix-based ACL rule to filter more bot traffic but less benign user traffic. Armbruster *et al.* [125] study the minimum set of nodes to deploy ingress/egress filtering to prevent IP spoofing on the Internet. Zhang *et al.* [126] model different filter placement strategies to study their efficacy when deploying fine-grained DDoS filters in transit networks. These solutions complement the above mitigation systems in more specific tasks.

Table 5. Selected DDoS mitigation solutions

	Main Idea	Example	Performance	Scalability	Deployability
Victim-Network	N/A	Out of scope	N/A	N/A	N/A
DDoS-Source (Sec. 4.3.1)	B.W. competition	Speak-up [109]	Fair	Medium	Low
	Rate limiting	CoDef [111]	Fair	High	Med
		D-WARD [11] COSSACK [90]	Good	High	Medium
	Whitelisting	Huici *et al.* [110]	Good	Low	Medium
In-Network: In-Line (Sec. 4.3.2)	Whitelisting	Ballani *et al.* [116]	Poor	Low	Low
	Software filtering	dFence [117]	Good	Medium	Medium
	Rate limiting	DefCOM [13]	Good	Medium	Medium
	Blacklisting	Jin *et al.* [112]	Poor	High	Low
		ATIF [12, 113] StopIt [114] SENSS [119]	Good	High	Medium
		Stellar [16]	Good	High	High
In-Network: Redirection (Sec. 4.3.3)	BGP poisoning	RAC [121]	Poor	Medium	Medium
	Overlay	CenterTrack [118]	Good	Low	Medium
	Software filtering	MiddlePolice [14] Bohatei [15]	Excellent	Medium	Medium

5 PREVIOUS DDOS SURVEYS

DDoS is a well-studied field, and there are a few highly cited surveys published over the years. These surveys laid a good foundation for us to understand the landscape of DDoS research, and they provide different classification methods for both attacks and defenses.

Mirkovic *et al.* [7], one of the most cited DDoS taxonomy, provides multiple classes for DDoS attacks and defenses. E.g., the survey classifies an attack by its degree of automation, victim type, source address validity, and it classifies defense solutions by their operational model, degree of cooperation, deployment location. The work serves as a general introduction to attack and defense from different perspectives but lacks detailed technical review of the latest research in each perspective. For example, there are many overlay-based prevention solutions, such as [75, 74, 79, 80], we need to understand why the particular assumption of a work does or does not hold.

Rather than providing many classes for attacks and defenses, Peng *et al.* [8] surveyed volumetric DDoS attacks that aim to inundate DDoS victims' inbound links. The survey provides a detailed workflow of attacks at different network layers, e.g., TCP SYN flood, HTTP flood, amplification attack. The work later uses four typical categories in security to summarize defense solutions: 1) attack prevention, 2) attack detection, 3) attack source identification, and 4) attack reaction. This survey was published in 2007 and does not capture many newly emerged attacks.

Similar to Peng *et al.* [8], Zargar *et al.* [9] surveyed volumetric attacks in 2013. The survey in [9] studies each defense solution by its defense location on the Internet: 1) source-end defense:

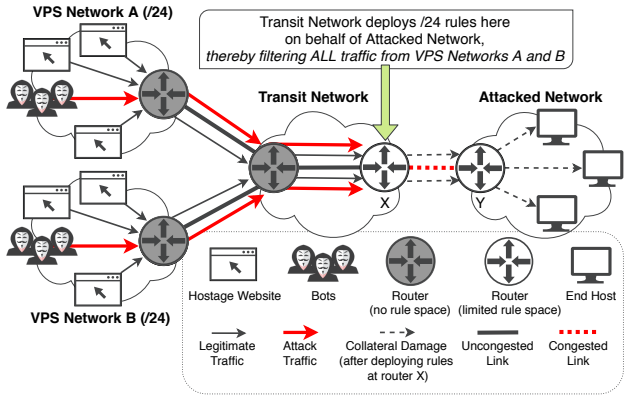

Fig. 13. The Catch-22 Attack [127]

defend victims at attack traffic source, 2) in-network defense: defend victims in the transit networks, and 3) victim-end defense: defend victims close to or at victim networks. The work suggests that a comprehensive defense solution needs to consider all defense locations to combat the latest attacks.

In this survey, we focus on the following: 1) a technical review of the modern DDoS, 2) a research of defense solutions that are practical/deployable without strong assumptions, and finally, 3) an evaluation of whether the existing DDoS defense research solutions are sufficient to combat the DDoS attacks.

6 CONCLUSION AND OPEN ISSUES

In this report, we introduced the background of DDoS attacks and defenses from a topological point of view, and examined the latest attacks and defense solutions. In the background section (Sec. 2), we classified existing DDoS attacks into two classes, *fast/slow-path* attacks, based on their intended resources. A fast-path attack aims to inundate the raw hardware processing capacity, e.g., the packet switching capacity of a router. A slow-path attack intends to overwhelm the computational, memory, or I/O resource of an end device anywhere on the Internet. We then classified existing DDoS defense research based on their strategies: *prevention*, *detection/classification*, and *mitigation* solutions. We also show readers the advantages and disadvantages of different defense locations on the Internet (Sec. 2.4). In Sec. 3, we studied the most common attacks in the wild, as well as the latest research in DDoS attacks, e.g., practical pulsing attacks [19, 20] and the stealthy link-flooding attacks [17, 18]. In Sec. 4, we examined the DDoS defense solutions from the past two decades. We reviewed the benefits and limitations of prevention solutions in Sec. 4.1, the status quo of DDoS detection/classification in Sec. 4.2, and the deployability of mitigation solutions in Sec. 4.3.

In reality, we rarely see ASes adapting the well-researched defense solutions for reasons such as deployment difficulty, system cost, maintenance cost. Perhaps, the biggest reason of all is that large-scale DDoS attacks are only relevant to the affected networks; these attacks rarely hit many networks at once. To motivate our network community to deploy simple but effective defense solutions, we believe one should show the community an estimated impact of modern DDoS attacks, To this end, we study a new DDoS attack, *the Catch-22 attack*, that leverages the available services on the Internet and imposes challenges on the existing defense solutions. In this attack, the attacker introduces a mitigation conundrum to victim networks with three steps:

- Acquire DDoS bots from networks that are meaningful to the victim networks, e.g., major cloud providers are crucial to residential networks.

- Launch any attack that saturates the link bandwidth of the victim networks.
- Rotate the attack targets for increasing the attack damage.

Since network routers rely on limited TCAM for high-speed packet processing, e.g., an up-to-date router can support 10s of thousands of ACL rules in its data plane, the TCAM space is not sufficient for mitigating large-scale attacks at host-level. Thus, the attacker is forcing the attacked networks to choose from: 1) filtering partial bot traffic but fails to mitigate the attack, or 2) filtering all bot traffic at the cost of losing connections to benign clients/servers. With either option, the attacked networks need to make difficult mitigation decisions.

Today, anyone can request (virtual) machines remotely from cloud providers. One can rent a machine with publicly-addressable IP for as little as $5 a month or $70 to launch an hour-long DDoS attack with 10,000 machines. Since the attack machines are co-located with other benign servers, a naive victim network may cause more harm to itself by mitigating the attack, if it installs coarse-grained traffic filters. Fig. 13 represents an example of the Catch-22 attack launched from cloud providers. With this attack, we examine fundamental vulnerabilities in today's DDoS defense infrastructure, as well as estimating/quantifying the attack damage if a Catch-22 attack is launched with the available resources on the Internet today. Finally, we list important DDoS research directions that are beneficial to the network community and the Internet:

Incentive study: the deployment of DDoS defense solutions is a long-standing problem. We hope to see more research that studies the requirements/features that would incentivize end users, edge networks, and transit networks to deploy their best-fit defense solutions at different defense strategies.

Sound detection/classification: while prevention solutions are crucial to DDoS defense, we cannot expect them to prevent all DDoS attacks. We should continue to work on DDoS detection/classification solutions that are not only able to detect/classify the latest DDoS attacks but also well-tested in realistic network environments, i.e., networks with real applications that generate traffic.

Alert handling: Incident handling is often overlooked in DDoS research. The DDoS research community rarely focus on what to do with the (false) alarms generated by detection/classification solutions. Even when a detection/classification solution claims to have a small false-positive rate, it could translate to an overwhelming amount of false alerts in large networks. We should provide answers to such a challenge for real-world deployment and defense automation.

Efficient mitigation: Lastly, should we find out it is impossible to push DDoS defense to the network edges, we need high-performance DDoS mitigation solutions that scales. E.g., a low-cost in-network DDoS mitigation system that allows users to deploy a huge amount of fine-grained traffic filters, yet not affecting the network throughput.

Realistic network environments: Finally, we realize it is often difficult, if not impossible, for many to find realistic network environments to evaluate their solutions, e.g., a high-speed detection solution for the CrossFire attack [18] that operates in transit networks. In fact, in our study, we did not find any sound detection/classification solution for the latest attacks. This issue challenges the credibility of many DDoS detection/classification research projects. Therefore, we should expect the research community to develop open platforms to enable such experiments, better yet, to replicate the experiments. One potential research direction for realizing such a platform is to employ time dilation technique in low-cost testbeds [128], and it would enable the testbeds to simulate large-scale network experiments with limited hardware capability at the cost of experiment time.

REFERENCES

[1] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, et al. 2017. Understanding the Mirai Botnet. In *26th USENIX Security Symposium (USENIX Security 17)*, 1093–1110.

[2] Memcached. 2018. Memcached – ReleaseNotes156. https://github.com/memcached/memcached/wiki/ReleaseNotes156.

[3] Catalin Cimpanu. 2019. 'Carpet-bombing' DDoS attack takes down South African ISP for an entire day.

[4] Warren Kumari and Danny McPherson. 2009. Remote Triggered Black Hole Filteringwith Unicast Reverse Path Forwarding (uRPF). Technical report. https://tools.ietf.org/html/rfc5635.

[5] Pedro Marques, Nischal Sheth, Robert Raszuk, Barry Greene, Jared Mauch, and Danny McPherson. 2009. Dissemination of Flow Specification Rules. Technical report. https://tools.ietf.org/html/rfc5575.

[6] Giovane C.M. Moura, Ricardo de O. Schmidt, John Heidemann, Wouter B. de Vries, Moritz Muller, Lan Wei, and Cristian Hesselman. 2016. Anycast vs. ddos: evaluating the november 2015 root dns event. In *Proceedings of the 2016 Internet Measurement Conference* (IMC '16), 255–270. DOI: 10.1145/2987443.2987446.

[7] Jelena Mirkovic and Peter Reiher. 2004. Taxonomy of DDoS Attackand DDoS Defense Mechanisms. *ACM SIGCOMM Computer Communication Review*, 34, 2, 39–53.

[8] Tao Peng, Christopher Leckie, and Kotagiri Ramamohanarao. 2007. Survey of Network-Based Defense Mechanisms Countering the DoS and DDoS Problems. *ACM Computing Surveys*, 39, 1, 3–es. DOI: 10.1145/1216370.1216373.

[9] Saman Taghavi Zargar, James Joshi, and David Tipper. 2013. A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks. *IEEE Communications Surveys and Tutorials*, 15, 4, 2046–2069. DOI: 10.1109/SURV.2013.031413.00127.

[10] Qiao Yan, F. Richard Yu, Qingxiang Gong, and Jianqiang Li. 2016. 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, 602–622. DOI: 10.1109/COMST.2015.2487361.

[11] Jelena Mirkovic, Gregory Prier, and Peter Reiher. 2002. Attacking DDoS at the Source. In *IEEE International Conference on Network Protocols*.

[12] Katerina Argyraki and David R. Cheriton. 2005. Active Internet Traffic Filtering: Real-time Response to Denial of Service Attacks. In *USENIX Annual Technical Conference*. Volume 38.

[13] George Oikonomou, Jelena Mirkovic, Peter Reiher, and Max Robinson. 2006. A Framework for a Collaborative DDoS Defense. In *Proceedings of the 22nd Annual Computer Security Applications Conference* (ACSAC'06), 33–42. DOI: 10.1109/ACSAC.2006.5.

[14] Zhuotao Liu, Hao Jin, Yih-Chun Hu, and Michael Bailey. 2016. 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* (CCS '16), 1268–1279. DOI: 10.1145/2976749.2978306.

[15] Seyed K. Fayaz, Yoshiaki Tobioka, Vyas Sekar, and Michael Bailey. 2015. Bohatei: Flexible and Elastic DDoS Defense. In *24th USENIX Security Symposium (USENIX Security 15)*, 817–832.

[16] Christoph Dietzel, Matthias Wichtlhuber, Georgios Smaragdakis, and Anja Feldmann. 2018. Stellar: Network Attack Mitigation Using Advanced Blackholing. In *Proceedings*

of the 14th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT'18), 152–164. DOI: 10.1145/3281411.3281413.

[17] Ahren Studer and Adrian Perrig. 2009. The Coremelt Attack. In *Computer Security – ESORICS 2009*, 37–52.

[18] Min Suk Kang, Soo Bum Lee, and Virgil D. Gligor. 2013. The Crossfire Attack. In *2013 IEEE Symposium on Security and Privacy*, 127–141. DOI: 10.1109/SP.2013.19.

[19] Yu-Ming Ke, Chih-Wei Chen, Hsu-Chun Hsiao, Adrian Perrig, and Vyas Sekar. 2016. Cicadas: congesting the internet with coordinated and decentralized pulsating attacks. In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security* (ASIA CCS '16), 699–710. DOI: 10.1145/2897845.2897866.

[20] Huasong Shan, Qingyang Wang, and Calton Pu. 2017. Tail Attacks on Web Applications. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security* (CCS'17), 1725–1739. DOI: 10.1145/3133956.3133968.

[21] Arpit Gupta, Laurent Vanbever, Muhammad Shahbaz, Sean P. Donovan, Brandon Schlinker, Nick Feamster, Jennifer Rexford, Scott Shenker, Russ Clark, and Ethan Katz-Bassett. 2014. SDX: A Software Defined Internet Exchange. In number 4. Volume 44, 551–562. DOI: 10.1145/2740070.2626300.

[22] Haopei Wang, Lei Xu, and Guofei Gu. 2015. FloodGuard: A DoS Attack Prevention Extension in Software-Defined Networks. In *2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 239–250. DOI: 10.1109/DSN.2015.27.

[23] Aggelos Lazaris, Daniel Tahara, Xin Huang, Erran Li, Andreas Voellmy, Y. Richard Yang, and Minlan Yu. 2014. Tango: Simplifying SDN Control with Automatic Switch Property Inference, Abstraction, and Optimization. In *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies* (CoNEXT '14), 199–212. DOI: 10.1145/2674005.2675011.

[24] Roland van Rijswijk-Deij, Anna Sperotto, and Aiko Pras. 2014. DNSSEC and Its Potential for DDoS Attacks: A Comprehensive Measurement Study. In *Proceedings of the 2014 Conference on Internet Measurement Conference* (IMC '14), 449–460. DOI: 10.1145/2663716.2663731.

[25] Malachi Kenney. 1998. It's the Ping o' Death Page! https://web.archive.org/web/19981206105844/http://www.sophist.demon.co.uk/ping/.

[26] Wesley M. Eddy. 2006. 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.

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

[28] Yang Xiang, Ke Li, and Wanlei Zhou. 2011. Low-Rate DDoS Attacks Detection and Traceback by Using New Information Metrics. *IEEE Transactions on Information Forensics and Security*, 6, 2, 426–437. DOI: 10.1109/TIFS.2011.2107320.

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

[30] Catalin Cimpanu. 2018. 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/.

[31] ServerHunter. 2019. Find a server. https://www.serverhunter.com.

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

[33] Paul Ferguson and Daniel Senie. 2000. Network Ingress Filtering: Defeating Denial-of-Service Attacks Which Employ IP Source Address Spoofing. Technical report. https://tools.ietf.org/html/rfc2827.

[34] CAIDA. 2019. State of IP Spoofing. https://spoofer.caida.org/summary.php.
[35] Sharon Goldberg. 2014. Why Is It Taking So Long to Secure Internet Routing? *Communications of the ACM*, 57, 10, 56–63.
[36] CAIDA. 2009. AS Relationships. https://www.caida.org/data/as-relationships.
[37] Haibin Sun, John C. S. Lui, and David K. Y. Yau. 2004. Defending against low-rate tcp attacks: dynamic detection and protection. In *Proceedings of the 12th IEEE International Conference on Network Protocols* (ICNP '04), 196–205.
[38] Aleksandar Kuzmanovic and Edward W. Knightly. 2003. Low-rate tcp-targeted denial of service attacks: the shrew vs. the mice and elephants. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (SIGCOMM '03), 75–86. DOI: 10.1145/863955.863966.
[39] Xiapu Luo and Rocky KC Chang. 2005. On a New Class of Pulsing Denial-of-Service Attacks and the Defense. In *Proceedings of the Network and Distributed System Security Symposium*.
[40] Ryan Rasti, Mukul Murthy, Nicholas Weaver, and Vern Paxson. 2015. Temporal lensing and its application in pulsing denial-of-service attacks. In *2015 IEEE Symposium on Security and Privacy*, 187–198. DOI: 10.1109/SP.2015.19.
[41] Qingyang Wang, Yasuhiko Kanemasa, Jack Li, Deepal Jayasinghe, Toshihiro Shimizu, Masazumi Matsubara, Motoyuki Kawaba, and Calton Pu. 2013. Detecting Transient Bottlenecks in n-Tier Applications through Fine-Grained Analysis. In *2013 IEEE 33rd International Conference on Distributed Computing Systems*, 31–40. DOI: 10.1109/ICDCS.2013.17.
[42] CERT Division. 1996. 1996 CERT Advisories. https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=496170.
[43] ntp.org. 2010. April 2010: DRDoS / Amplification Attack using ntpdc monlist command. http://support.ntp.org/bin/view/Main/SecurityNotice.
[44] RSnake. 2009. Slowloris HTTP DoS. https://web.archive.org/web/20150426090206/http://ha.ckers.org/slowloris.
[45] Cloudflare. 2019. What is a DNS Flood? https://www.cloudflare.com/learning/ddos/dns-flood-ddos-attack/.
[46] Cloudflare. 2019. What is an HTTP flood DDoS attack? https://www.cloudflare.com/learning/ddos/http-flood-ddos-attack/.
[47] Mark Handley and Eric Rescorla. 2006. Internet Denial-of-Service Considerations. Technical report. https://tools.ietf.org/html/rfc4732.
[48] Akamai. 2019. Why Akamai Cloud Security for DDoS Protection? https://www.akamai.com/us/en/products/security/ddos-protection-service.jsp.
[49] Cloudflare. 2019. Advanced DDoS Attack Protection. https://www.cloudflare.com/ddos.
[50] Lily Hay Newman. 2018. GitHub Survived the Biggest DDoS Attack Ever Recorded. https://www.wired.com/story/github-ddos-memcached.
[51] Mina Tahmasbi Arashloo, Alexey Lavrov, Manya Ghobadi, Jennifer Rexford, David Walker, and David Wentzlaff. 2020. Enabling Programmable Transport Protocols in High-Speed NICs. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*.
[52] Tom Killalea. 2000. Recommended Internet Service Provider Security Services and Procedures. Technical report. https://tools.ietf.org/html/rfc3013.
[53] Kihong Park and Heejo Lee. 2001. On the Effectiveness of Route-Based Packet Filtering for Distributed DoS Attack Prevention in Power-Law Internets. *SIGCOMM Computer Communication Review*, 31, 4, 15–26. DOI: 10.1145/964723.383061.

[54] Jelena Mirkovic, Erik Kline, and Peter Reiher. 2017. Resect: self-learning traffic filters for ip spoofing defense. In *Proceedings of the 33rd Annual Computer Security Applications Conference* (ACSAC 2017), 474–485. DOI: 10.1145/3134600.3134644.

[55] Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford. 2003. Captcha: using hard ai problems for security. In *Advances in Cryptology — EUROCRYPT 2003*, 294–311.

[56] Ramakrishna Gummadi, Hari Balakrishnan, Petros Maniatis, and Sylvia Ratnasamy. 2009. Not-a-bot: improving service availability in the face of botnet attacks. In *USENIX Symposium on Networked Systems Design and Implementation (NSDI'09)*.

[57] Trusted Computing Group. 2011. Trusted Platform Module Main Specification. https://trustedcomputinggroup.org/resource/tpm-main-specification.

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

[59] Michael J. Freedman, Karthik Lakshminarayanan, and David Mazières. 2006. OASIS: Anycast for Any Service. In *Networked Systems Design and Implementation* (NSDI'06).

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

[61] Ashley Flavel, Pradeepkumar Mani, David Maltz, Nick Holt, Jie Liu, Yingying Chen, and Oleg Surmachev. 2015. FastRoute: A Scalable Load-Aware Anycast Routing Architecture for Modern CDNs. In *12th USENIX Symposium on Networked Systems Design and Implementation* (NSDI'15), 381–394.

[62] Henri Maxime Demoulin, Tavish Vaidya, Isaac Pedisich, Bob Dimaiolo, et al. 2018. DeDoS: Defusing DoS with Dispersion Oriented Software. In *Proceedings of the 34th Annual Computer Security Applications Conference* (ACSAC '18), 712–722. DOI: 10.1145/3274694.3274727.

[63] Lin Jin, Shuai Hao, Haining Wang, and Chase Cotton. 2018. Your Remnant Tells Secret: Residual Resolution in DDoS Protection Services. In *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 362–373. DOI: 10.1109/DSN.2018.00046.

[64] Jafar Haadi Jafarian, Ehab Al-Shaer, and Qi Duan. 2012. Openflow random host mutation: transparent moving target defense using software defined networking. In *Proceedings of the First Workshop on Hot Topics in Software Defined Networks* (HotSDN '12), 127–132. DOI: 10.1145/2342441.2342467.

[65] Quan Jia, Kun Sun, and Angelos Stavrou. 2013. MOTAG: Moving Target Defense against Internet Denial of Service Attacks. In *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*, 1–9. DOI: 10.1109/ICCCN.2013.6614155.

[66] Quan Jia, Huangxin Wang, Dan Fleck, Fei Li, Angelos Stavrou, and Walter Powell. 2014. Catch Me if You Can: A Cloud-Enabled DDoS Defense. In *2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, 264–275. DOI: 10.1109/DSN.2014.35.

[67] Sridhar Venkatesan, Massimiliano Albanese, Kareem Amin, Sushil Jajodia, and Mason Wright. 2016. A Moving Target Defense Approach to MitigateDDoS Attacks against Proxy-Based Architectures. In *2016 IEEE Conference on Communications and Network Security (CNS)*, 198–206. DOI: 10.1109/CNS.2016.7860486.

[68] Saptarshi Debroy, Prasad Calyam, Minh Nguyen, Allen Stage, and Vladimir Georgiev. 2016. Frequency-Minimal Moving Target Defense using Software-Defined Networking. In *2016 International Conference on Computing, Networking and Communications (ICNC)*, 1–6. DOI: 10.1109/ICCNC.2016.7440635.

[69] Yuquan Shan, George Kesidis, and Daniel Fleck. 2017. Cloud-side shuffling defenses against ddos attacks on proxied multiserver systems. In *Proceedings of the 2017 on Cloud Computing Security Workshop* (CCSW '17), 1–10. DOI: 10.1145/3140649.3140650.

[70] Jin Bum Hong and Dong Seong Kim. 2016. Assessing the effectiveness of moving target defenses using security models. *IEEE Transactions on Dependable and Secure Computing*, 13, 2, 163–177. DOI: 10.1109/TDSC.2015.2443790.

[71] Mason Wright, Sridhar Venkatesan, Massimiliano Albanese, and Michael P. Wellman. 2016. Moving Target Defense against DDoS Attacks: An Empirical Game-Theoretic Analysis. In *Proceedings of the 2016 ACM Workshop on Moving Target Defense* (MTD '16), 93–104. DOI: 10.1145/2995272.2995279.

[72] Sherif Khattab, Sameh Gobriel, Rami Melhem, and D. Mosse. 2008. Live Baiting for Service-Level DoS Attackers. In *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*, 171–175. DOI: 10.1109/INFOCOM.2008.43.

[73] Abraham Yaar, Adrian Perrig, and Dawn Song. 2004. SIFF: A Stateless Internet Flow Filterto Mitigate DDoS Flooding Attacks. In *IEEE Symposium on Security and Privacy*, 130–143. DOI: 10.1109/SECPRI.2004.1301320.

[74] A. D. Keromytis, V. Misra, and D. Rubenstein. 2004. SOS: An Architecture for Mitigating DDoS Attacks. *IEEE Journal on Selected Areas in Communications*, 22, 1, 176–188. DOI: 10.1109/JSAC.2003.818807.

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

[76] William G. Morein, Angelos Stavrou, Debra L. Cook, Angelos D. Keromytis, Vishal Misra, and Dan Rubenstein. 2003. Using Graphic Turing Tests to Counter Automated DDoS Attacks against Web Servers. In *Proceedings of the 10th ACM Conference on Computer and Communications Security* (CCS '03), 8–19. DOI: 10.1145/948109.948114.

[77] Angelos Stavrou and Angelos D. Keromytis. 2005. Countering dos attacks with stateless multipath overlays. In *Proceedings of the 12th ACM Conference on Computer and Communications Security* (CCS '05), 249–259. DOI: 10.1145/1102120.1102153.

[78] Alan Ford, Costin Raiciu, Mark Handley, Olivier Bonaventure, and Christoph Paasch. 2019. TCP Extensions for Multipath Operation with Multiple Addresses. Technical report. https://www.ietf.org/id/draft-ietf-mptcp-rfc6824bis-18.txt.

[79] Colin Dixon, Thomas Anderson, and Arvind Krishnamurthy. 2008. Phalanx: Withstanding Multimillion-Node Botnets. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation* (NSDI'08), 45–58.

[80] Kübra Kalkan and Fatih Alagöz. 2016. A distributed filtering mechanism against DDoS attacks: ScoreForCore. *Computer Networks*, 108, 199 –209. DOI: https://doi.org/10.1016/j.comnet.2016.08.023.

[81] Hsu-Chun Hsiao, Tiffany Hyun-Jin Kim, Sangjae Yoo, Xin Zhang, Soo Bum Lee, Virgil Gligor, and Adrian Perrig. 2013. STRIDE: Sanctuary Trail – Refuge from Internet DDoS Entrapment. In *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security* (ASIA CCS '13), 415–426. DOI: 10.1145/2484313.2484367.

[82] Cristina Basescu, Raphael M Reischuk, Pawel Szalachowski, Adrian Perrig, Yao Zhang, Hsu-Chun Hsiao, Ayumu Kubota, and Jumpei Urakawa. 2016. SIBRA: Scalable Internet Bandwidth Reservation Architecture. In *Network and Distributed System Security Symposium* (NDSS'16).

[83] Xin Zhang, Hsu-Chun Hsiao, Geoffrey Hasker, Haowen Chan, Adrian Perrig, and David G Andersen. 2011. SCION: Scalability, Control, and Isolation on Next-Generation Networks. In *2011 IEEE Symposium on Security and Privacy*, 212–227. DOI: 10.1109/SP.2011.45.

[84] Anukool Lakhina, Mark Crovella, and Christiphe Diot. 2004. Characterization of Network-Wide Anomalies in Traffic Flows. In *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement* (IMC '04), 201–206. DOI: 10.1145/1028788.1028813.

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

[86] Xin Li, Fang Bian, Mark Crovella, Christophe Diot, Ramesh Govindan, Gianluca Iannaccone, and Anukool Lakhina. 2006. Detection and Identification of Network Anomalies Using Sketch Subspaces. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement* (IMC '06), 147–152. DOI: 10.1145/1177080.1177099.

[87] Fernando Silveira, Christophe Diot, Nina Taft, and Ramesh Govindan. 2010. ASTUTE: Detecting a Different Class of Traffic Anomalies. *SIGCOMM Computer Communication Review*, 40, 4, 267–278. ISSN: 0146-4833. DOI: 10.1145/1851275.1851215.

[88] Alefiya Hussain, John Heidemann, and Christos Papadopoulos. 2003. A Framework for Classifying Denial of Service Attacks. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (SIGCOMM '03), 99–110. DOI: 10.1145/863955.863968.

[89] Alefiya Hussain, John Heidemann, and Christos Papadopoulos. 2006. Identification of Repeated Denial of Service Attacks. In *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, 1–15. DOI: 10.1109/INFOCOM.2006.126.

[90] Christos Papadopoulos, Robert Lindell, John Mehringer, Alefiya Hussain, and Ramesh Govindan. 2003. COSSACK: Coordinated Suppression of Simultaneous Attacks. In *Proceedings DARPA Information Survivability Conference and Exposition*. Volume 1, 2–13.

[91] Dimitris Gavrilis and Evangelos Dermatas. 2005. Real-Time Detection of Distributed Denial-of-Service Attacks Using RBF Networks and Statistical Features. *Computer Networks*, 48, 2, 235–245.

[92] Opeyemi Osanaiye, Haibin Cai, Kim-Kwang Raymond Choo, Ali Dehghantanha, Zheng Xu, and Mqhele Dlodlo. 2016. Ensemble-based multi-filter feature selection method for DDoS detection in cloud computing. *EURASIP Journal on Wireless Communications and Networking*, 2016, 1. DOI: 10.1186/s13638-016-0623-3.

[93] George Nychis, Vyas Sekar, David G. Andersen, Hyong Kim, and Hui Zhang. 2008. An empirical evaluation of entropy-based traffic anomaly detection. In *Proceedings of the 8th ACM SIGCOMM Conference on Internet Measurement* (IMC '08), 151–156. DOI: 10.1145/1452520.1452539.

[94] Yi Xie and Shun-Zheng Yu. 2009. Monitoring the application-layer ddos attacks for popular websites. *IEEE/ACM Transactions on Networking*, 17, 1, 15–25. DOI: 10.1109/TNET.2008.925628.

[95] Sunny Behal and Krishan Kumar. 2017. Detection of DDoS Attacks and Flash Events Using Novel Information Theory Metrics. *Computer Networks*, 116, 96–110. DOI: 10.1016/j.comnet.2017.02.015.

[96] Kejie Lu, Dapeng Wu, Jieyan Fan, Sinisa Todorovic, and Antonio Nucci. 2007. Robust and efficient detection of ddos attacks for large-scale internet. *Computer Networks*, 51, 18, 5036–5056. DOI: https://doi.org/10.1016/j.comnet.2007.08.008.

[97] Supranamaya Ranjan, Ram Swaminathan, Mustafa Uysal, and Edward Knightly. 2006. DDoS-Resilient Scheduling to Counter Application Layer Attacks Under Imperfect Detection. In *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, 1–13. DOI: 10.1109/INFOCOM.2006.127.

[98] Supranamaya Ranjan, Ram Swaminathan, Mustafa Uysal, Antonio Nucci, and Edward Knightly. 2009. DDoS-Shield: DDoS-Resilient Scheduling to Counter Application Layer Attacks. *IEEE/ACM Transactions on Networking*, 17, 1, 26–39. DOI: 10.1109/TNET.2008.926503.

[99] Yehuda Afek, Anat Bremler-Barr, and Shir Landau Feibish. 2019. Zero-Day Signature Extraction for High-Volume Attacks. *IEEE/ACM Transactions on Networking*, 27, 2, 691–706. DOI: 10.1109/TNET.2019.2899124.

[100] Vyas Sekar, Nick Duffield, Oliver Spatscheck, Jacobus van der Merwe, and Hui Zhang. 2006. LADS: Large-scale Automated DDoS Detection System. *USENIX Annual Technical Conference, General Track*, 171–184.

[101] Changwang Zhang, Zhiping Cai, Weifeng Chen, Xiapu Luo, and Jianping Yin. 2012. Flow level detection and filtering of low-rate ddos. *Computer Networks*, 56, 15, 3417–3431. DOI: 10.1016/j.comnet.2012.07.003.

[102] Jérôme François, Issam Aib, and Raouf Boutaba. 2012. FireCol: A Collaborative Protection Network for the Detection of Flooding DDoS Attacks. *IEEE/ACM Transactions on Networking*, 20, 6, 1828–1841. DOI: 10.1109/TNET.2012.2194508.

[103] Yang Xu and Yong Liu. 2016. DDoS Attack Detection Under SDN Context. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, 1–9. DOI: 10.1109/INFOCOM.2016.7524500.

[104] Jing Zheng, Qi Li, Guofei Gu, Jiahao Cao, David K.Y. Yau, and Jianping Wu. 2018. Realtime DDoS Defense Using COTS SDN Switches via Adaptive Correlation Analysis. *IEEE Transactions on Information Forensics and Security*, 13, 7, 1838–1853. DOI: 10.1109/TIFS.2018.2805600.

[105] Mohan Dhawan, Rishabh Poddar, Kshiteej Mahajan, and Vijay Mann. 2015. SPHINX: Detecting Security Attacks in Software-Defined Networks. In *Network and Distributed System Security Symposium*.

[106] Min Suk Kang, Virgil D Gligor, and Vyas Sekar. 2016. SPIFFY: Inducing Cost-Detectability Tradeoffs for Persistent Link-Flooding Attacks. *Network and Distributed System Security Symposium*.

[107] MyungKeun Yoon, Tao Li, Shigang Chen, and Jih-Kwon Peir. 2011. Fit a Compact Spread Estimator in Small High-Speed Memory. *IEEE/ACM Transactions on Networking*, 19, 5, 1253–1264. ISSN: 1063-6692. DOI: 10.1109/TNET.2010.2080285.

[108] Minlan Yu, Lavanya Jose, and Rui Miao. 2013. Software defined traffic measurement with opensketch. In *Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation* (NSDI'13), 29–42.

[109] Michael Walfish, Mythili Vutukuru, Hari Balakrishnan, David Karger, and Scott Shenker. 2010. DDoS Defense by Offense. *ACM Transactions on Computer Systems*, 28, 1, 1–54. DOI: 10.1145/1731060.1731063.

[110] Felipe Huici and Mark Handley. 2007. An Edge-to-Edge Filtering Architecture Against DoS. *SIGCOMM Computer Communication Review*, 37, 2. DOI: 10.1145/1232919.1232924.

[111] Soo Bum Lee, Min Suk Kang, and Virgil D. Gligor. 2013. CoDef: Collaborative Defense against Large-Scale Link-Flooding Attacks. In *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies* (CoNEXT '13), 417–428. DOI: 10.1145/2535372.2535398.

[112] Cheng Jin, Haining Wang, and Kang G. Shin. 2003. Hop-Count Filtering: An Effective Defense against Spoofed DDoS Traffic. In *Proceedings of the 10th ACM Conference on Computer and Communications Security* (CCS '03), 30–41. DOI: 10.1145/948109.948116.

[113] Katerina Argyraki and David R. Cheriton. 2009. Scalable Network-Layer Defense Against Internet Bandwidth-Flooding Attacks. *IEEE/ACM Transactions on Networking*, 17, 4, 1284–1297. DOI: 10.1109/TNET.2008.2007431.

[114] Xin Liu, Xiaowei Yang, and Yanbin Lu. 2008. To Filter or to Authorize: Network-Layer DoS Defense against Multimillion-Node Botnets. *SIGCOMM Computer Communication Review*, 38, 4, 195–206. DOI: 10.1145/1402946.1402981.

[115] Xin Liu, Ang Li, Xiaowei Yang, and David Wetherall. 2008. Passport: secure and adoptable source authentication. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation* (NSDI'08), 365–378.

[116] Hitesh Ballani and Yatin Chawathe. 2005. Off by Default! *ACM Workshop on Hot Topics in Networks*.

[117] Ajay Mahimkar, Jasraj Dange, Vitaly Shmatikov, Harrick Vin, and Yin Zhang. 2007. dFence: Transparent Network-based Denial of Service Mitigation. *4th USENIX Symposium on Networked Systems Design & Implementation*. NSDI'07.

[118] Robert Stone. 2000. Centertrack: An IP Overlay Network for Tracking DoS Floods. In *Proceedings of the 9th Conference on USENIX Security Symposium* (SSYM'00), 15.

[119] Sivaramakrishnan Ramanathan, Jelena Mirkovic, Minlan Yu, and Ying Zhang. 2018. SENSS Against Volumetric DDoS Attacks. In *Proceedings of the 34th Annual Computer Security Applications Conference* (ACSAC'18), 266–277. DOI: 10.1145/3274694.3274717.

[120] Jessica Steinberger, Benjamin Kuhnert, Anna Sperotto, Harald Baier, and Aiko Pras. 2016. Collaborative DDoS Defense using Flow-based Security Event Information. In *IEEE/IFIP Network Operations and Management Symposium*, 516–522. DOI: 10.1109/NOMS.2016.7502852.

[121] Jared M. Smith and Max Schuchard. 2018. Routing Around Congestion: Defeating DDoS Attacks and Adverse Network Conditions via Reactive BGP Routing. *IEEE Symposium on Security and Privacy*.

[122] Muoi Tran, Min Suk Kang, Hsu-Chun Hsiao, Wei-Hsuan Chiang, Shu-Po Tung, and Yu-Su Wang. 2019. On the feasibility of rerouting-based ddos defenses. In *IEEE Symposium on Security and Privacy*.

[123] Fabio Soldo, Athina Markopoulou, and Katerina Argyraki. 2009. Optimal Filtering of Source Address Prefixes: Models and Algorithms. In *IEEE INFOCOM 2009*, 2446–2454. DOI: 10.1109/INFCOM.2009.5062172.

[124] Fabio Soldo, Katerina Argyraki, and Athina Markopoulou. 2012. Optimal Source-Based Filtering of Malicious Traffic. *IEEE/ACM Transactions on Networking*, 20, 2, 381–395. DOI: 10.1109/TNET.2011.2161615.

[125] Benjamin Armbruster, J. Cole Smith, and Kihong Park. 2007. A packet filter placement problem with application to defense against spoofed denial of service attacks. *European Journal of Operational Research*, 176, 2, 1283–1292. DOI: https://doi.org/10.1016/j.ejor.2005.09.031.

[126] Mingwei Zhang, Lumin Shi, Devkishen Sisodia, Jun Li, and Peter Reiher. 2019. On Multi-Point, In-Network Filtering of Distributed Denial-of-Service Traffic. In *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 180–188.

[127] Lumin Shi, Devkishen Sisodia, Mingwei Zhang, Jun Li, Alberto Dainotti, and Peter Reiher. 2019. The Catch-22 Attack (Poster). In *Annual Computer Security Applications Conference (ACSAC)*.

[128] Rohit Chormale. 2019. Virtual time via time-dilation. https://github.com/mininet/mininet/wiki/Ideas#virtual-time-via-time-dilation.