DynaShield: Reducing the Cost of DDoS Defense using Cloud Services

Shengbao Zheng, Xiaowei Yang
Duke University

Abstract

Fueled by IoT botnets and DDoS-as-a-Service tools, distributed denial of service (DDoS) attacks have reached record high volumes. Although there exist DDoS protection services, they can be costly for small organizations as well as individual users. In this paper, we present a low-cost DDoS solution, DynaShield, which a user can deploy at common cloud service providers. DynaShield employs three techniques to reduce cost. First, it uses an on-demand model. A server dynamically updates its DNS record to redirect clients' traffic to DynaShield when it is under attack, avoiding paying for cloud services during peacetime. Second, DynaShield combines serverless functions and elastic servers provided by cloud providers to auto-scale to large attacks without over-provisioning. Third, DynaShield uses cryptocurrency puzzles as proof of work. The coin mining profit can further offset a protected server's cloud service charges. Our preliminary evaluation suggests that DynaShield can cost as little as a few dollars per month to prevent an organization from common DDoS attacks.

1 Introduction

Distributed Denial of Service (DDoS) attacks are a long time security threat. Except for a few Internet giants, anyone can become a DDoS victim. The emergence of DDoS-As-a-Service and IoT botnets makes launching such attacks increasingly easy and cheap. According to [2], botnet creators nowadays sell DDoS-As-a-Service with guaranteed 290Gbps volume at a cost as low as $20/month. In contrast, defending against such attacks can be painfully expensive for many organizations and Internet users. In 2015, Rutgers university suffered a sustained series of DDoS attacks. The university was forced to spend more than $300K in consultation and consequently increased its tuition and fees by 2.3% to cover its cyber-security cost [16, 21].

Both industry and academia have come to rescue. Today, there exist a variety of DDoS protection products and services [1, 3, 6–8, 14, 17] as well as a plethora of academic proposals to make the Internet DDoS resistant [22, 24, 25, 27, 30–36, 38–45]. However, the industry solutions often charge a non-negilible fee and can be a financial burden for small or non-profit organizations or individual Internet users. For instance, the DDoS protection service provider Cloudflare [8] charges a minimal flat fee of $200/month per business customer, and an additional $5 per million good requests destined to the customer. The academic solutions often require changes to routers or the Internet architecture, and cannot be readily deployed on Today's Internet.

In this work, we explore how an organization or an individual user can protect themselves from DDoS attacks at a low cost using today's technologies. To this end, we outline a solution, DynaShield, which provides on-demand and low-cost DDoS protection using cloud services. DynaShield uses common cloud services to build a protective shield for a DDoS victim. We are attracted to cloud services because 1) the current pricing for cloud services is more favorable than that of comparable DDoS protection services; and 2) most cloud providers already have large-scale distributed networks that are resilient to raw bandwidth flooding attacks, and they scrub off network layer and transport layer attack traffic for free [6, 7, 14].

DynaShield uses three techniques to reduce its cost of using cloud services. First, it exploits the intermittent nature of DDoS attacks, and only uses cloud services on demand. A server redirects its traffic to DynaShield after it encounters an attack using dynamic DNS updates, eliminating the cloud service charges during peacetime. Second, DynaShield uses a combination of Function-as-a-Service (FaaS) and Infrastructure-as-a-Service (IaaS) to auto-scale to large attacks while keeping its cost low. Third, DynaShield uses cryptocurrency as Proof-of-Work to limit how fast malicious clients can send application-layer requests. The profit from coin mining further offsets its cloud service cost.

To facilitate deployment, DynaShield uses an existing BGP feature Flowspec [37] to prevent attack traffic from directly reaching a protected server (§ 3.6). It does not require any client side modification to protect web services. We currently design DynaShield to only protect web services, but it is our future work to extend it to protect other services (§ 6).

DynaShield has a main limitation. It assumes the cloud provider it builds on, the DNS service, the Internet backbone, and the Internet service provider (ISP) a protected server uses have sufficient network capacity to withstand raw bandwidth flooding attacks. Therefore, it is not suitable for defending against large-scale DDoS attacks that can take down a cloud provider, a DNS service, or an ISP.

We evaluate DynaShield's on-demand DNS redirection latency, and compare its latency inflation and cost with those of a commerical service: Cloudflare. The results show that DynaShield's DNS redirection latency is almost the same as the TTL of a server's DNS record. A server that sets a short TTL on its DNS record (e.g., 30s) will experience a short DNS redirection latency. DynaShield's latency inflation is similar to that of Cloudflare and is on average less than 22% of the direct connection latency from a client to a server. For a web server `www.duke.edu` we study, it reduces the DDoS protection cost from a few hundred dollars per month charged by Cloudflare to a few dollars. This result suggests that DynaShield can be a low-cost DDoS solution. It is our future work to build DynaShield and evaluate its effectiveness to combat various DDoS attacks.

2 Overview

In this section, we describe DynaShield's design assumptions, its design goals, and the adversary model.

2.1 Assumptions

Cloud Service Model We assume two main types of cloud services: IaaS and FaaS. With IaaS, a user specifies a Virtual Machine (VM) instance's compute resource provision and manages the start and stop of the first VM instance by himself. A service built using IaaS scales at the granularity of a VM. That is, a service deploys more unit machines when the load on the current VM instances is high.

In contrast, FaaS, i.e. serverless functions like AWS Lambda, enables a user to run an application without provisioning or managing the needed computing resources by himself. The user implements an application using one or more functions. A service built using serverless functions scales at the granularity of a function. That is, a cloud provider increases the number of function instances when the load is high.

Pricing Model Both IaaS and FaaS have a pay-for-use model. A VM provided by IaaS is charged based on the VM's compute resource provision and the time it runs. Serverless functions are charged based on how many times a function is invoked as well as its resource consumption.

Well-Connected Infrastructure We assume the cloud provider DynaShield builds on, a protected server's DNS provider, the Internet backbone, and the ISP a protected server connects to have sufficient bandwidth to withstand most raw bandwidth flooding DDoS attacks. We make this assumption because commercial DDoS protection services such as Cloudflare and Akamai all rely on the Internet backbone, their own networks, and DNS services having sufficient bandwidth to withstand the raw bandwidth flooding attacks.

2.2 Adversary Model

Attack duration and frequency We assume DDoS attacks are intermittent. According to the DDoS reports of the last two years from Incapsula [15], Kaspersky Lab [11], and Akamai [20], a DDoS attack usually has a short duration. Specifically, the report from Kaspersky Lab [11] shows that among all DDoS attacks they investigated in the first two quarters of 2018, 80.73% and 69.49% of them are shorter than four hours respectively. Incapsula [15] shows that for the last three months of 2017, a targeted DDoS victim suffered on average 2.9 attacks per month. These results are consistent with the financial incentives of attackers, as launching persistent attacks requires them to spend more money on purchasing the attack infrastructure. Launching intermittent attacks can cause service disruptions at a low cost, and also prevents the exposure of the attack infrastructure.

Types of DDoS attacks We consider all layers of DDoS flooding attacks, including both infrastructure and application layer DDoS attacks. We allow all attackers to collude and synchronize their floods. Attacks may also try to bypass DynaShield and access a protected server directly to launch an attack. However, we assume that there is no on-path attacker, as such an attacker can simply discard all traffic it receives to launch DDoS attacks.

Ability to Detect DDoS Attacks We assume a server can detect the onset of a DDoS attack, thereby invoking DynaShield to protect itself. It is outside the scope of this work to discuss how to detect DDoS attacks. A server can use its load, traffic pattern, or existing DDoS attack detection products [13] for this purpose.

2.3 Design Goals

Effective DDoS protection We aim to make DynaShield effectively protect a server from infrastructure and application layer DDoS attacks. When a DDoS attack occurs, a protected server does not crash, and can serve legitimate clients' requests according to its service policy.

Low cost We aim to make the cost of DynaShield significantly lower than that of purchasing services from companies like Cloudflare and Akamai.

Deployable We aim to make DynaShield immediately deployable on the Internet without modifying routers or clients.

Flexible We aim to enable a protected server to deploy flexible policies to provide differentiated services to different clients during attack times. For instance, a server may degrade services to suspected malicious clients, or treat all indistinguishable clients fairly.

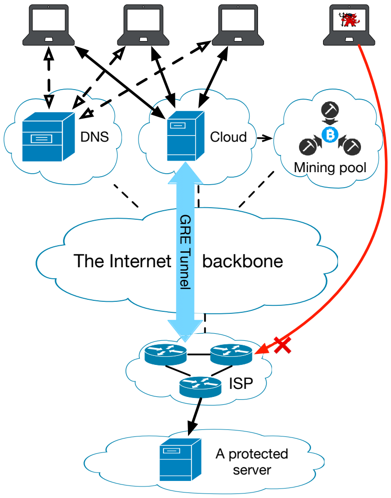

Figure 1: **The DynaShield architecture.**

3 Design

In this section, we describe the design of DynaShield.

3.1 The DynaShield Architecture

Figure 1 shows the high-level DynaShield architecture. It has five participating components: a protected server's DNS service, DynaShield's serverless functions and VM instances running on the cloud, existing cryptocurrency mining pools, an ISP's filtering routers, and a protected server.

When a server detects the onset of an DDoS attack, it redirects its traffic to DynaShield by updating its DNS records to point to DynaShield's IP addresses, and informs its upstream ISPs using BGP Flowspec [37] to install router filters that only allow traffic from DynaShield to reach itself. DynaShield forwards the traffic to the server using a Generic Routing Encapsulation (GRE) tunnel [29] (§ 3.6).

DynaShield is a capability-based DDoS defense architecture [23]. Before a client can send to a server, it must obtain permissions to send. A server returns permissions to send as capability tokens and a client carries those tokens in its subsequent requests. Different from previous work [35, 38, 43–45], DynaShield uses application-layer capabilities to avoid modifying a client. DynaShield acts like an HTTP/HTTPs proxy, terminating and forwarding a client's HTTP/HTTPs requests to a protected server. DynaShield treats the first HTTP request from a client as the client's initial request. If the server accepts the client's request, it will return a capability token using an HTTP cookie protected by a secret key known only to itself and DynaShield. A client's subsequent requests will carry the HTTP cookie. DynaShield discards unwanted traffic without a valid cookie.

3.2 On-demand DDoS Protection

DynaShield mitigates DDoS attacks on demand. A protected server operates in two states: the peacetime state and the under-threat state. The server enters the under-threat state when it detects the onset of a DDoS attack. When a server enters the under-threat state, DynaShield will be triggered either automatically or manually by the system administrator. We use DNS updates to redirect the server's traffic to DynaShield. DynaShield will also wake up its first VM instance at its cloud provider.

3.3 Initial Requests

DynaShield uses serverless functions to handle the initial request from a client. This design allows DynaShield to automatically scale to a large volume of attack traffic without knowing the size of the attack ahead of time. Additionally, the start time of a serverless function is much shorter than that of an elastic server [5]. So this design can avoid service disruption when a server transitions from the peacetime state to the under-threat state.

The initial requests may come from malicious clients and overwhelm a server. To address this problem, DynaShield uses computational puzzles derived from cryptocurrency as Proof-of-Work to limit the initial request rate while enforcing computational fairness among all clients. DynaShield could use traditional computational puzzles as in previous work [27, 38]. However, traditional puzzles cause similar amount of work at the client side but offers no real value to the server.

Differently, DynaShield joins a mining pool as a miner. It divides the cryptocurrency puzzles received from the mining pool into many easier pieces. Since cryptocurrency puzzles have real values, DynaShield can aggregate the puzzle solutions from multiple clients to obtain mining profit. Without specific mentioning, we also refer to these easier puzzle pieces as puzzles in the rest of the paper.

In our initial design, DynaShield returns a puzzle in response to a client's initial request as a JavaScript applet, which includes the puzzle solving code. A client can solve the puzzle by running the code without installing any additional application. When a client solves a puzzle, the applet will submit the solution to DynaShield. DynaShield verifies the puzzle solution. If the solution is correct, it forwards the client's initial request to the server.

DynaShield can adjust the difficulty of a puzzle to adjust the request rate it forwards to the protected server. It is our future work to study how to adjust the puzzle difficulties to match an expected incoming request rate set by a server.

When a server receives an initial request forwarded by DynaShield, it will decide whether to accept the client's connection. If it does, it will return a cryptographically-derived capability token to the client. Malicious clients cannot forge capability tokens. A client's subsequent requests will carry the capability tokens as proof of authorization to send.

3.4 Authorized Requests

DynaShield uses VM instances provided by its cloud provider to forward authorized requests. It returns a Javascript applet to a client after it verifies the client's puzzle solution. This applet redirects the client's subsequent requests to its VM instances. DynaShield's VM instances verify the capability tokens carried in a client's requests, and forward the requests with valid capability tokens to the server.

DynaShield can use serverless functions to verify authorized requests, but using VMs avoids per request charges. Since a server can control the authorized request rate by controlling how capability tokens are returned to clients, it can provision the VM instances to match the expected authorized request rate, thereby avoiding overprovisioning the VM instances and paying unnecessary cloud charges.

3.5 Blacklisting Non-Compliant Clients

A malicious client may request a puzzle without solving it or send an invalid capability token to DynaShield. Any request that reaches DynaShield consumes cloud resources, incurring cost to a server. To avoid this undesirable cost, DynaShield keeps a record of clients who repeatedly fail to solve a puzzle or send an invalid capability token in a blacklist, and requests the cloud provider to filter their traffic at the network layer for a period of time.

3.6 Preventing Bypassing Attacks

Attackers can bypass DynaShield by directly sending traffic to a server's IP address. One possible solution is to assign a protected server a different IP address and keep it as a secret when it is in the under-threat state. However, this design has the drawback that a server must renumber between the under-threat and the peacetime state.

As deployability is a main goal, we use a different mechanism to address the bypassing vulnerability. When DynaShield is launched, it establishes a GRE tunnel to the protected server, but uses a spoofed secret source IP address $secretIP_{DynaShield}$ as the tunnel's source IP address.

A protected server will request its upstream ISP to install a filter that only allows traffic from this GRE tunnel, and discards all traffic sent to its IP address. DynaShield will forward all traffic to the protected server via this GRE tunnel. Note that the return traffic from the server to DynaShield need not go through the tunnel: a server can send the return traffic to DynaShield's public IP addresses directly.

With this design, an attacker cannot bypass DynaShield to flood the server unless it correctly guesses the tunnel's source IP address. The probability is only around 2^{-32}. DynaShield can dynamically change the tunnel's source IP address over time and update router filters to defend against persistent attacks that scan the entire IPv4 address space.

4 Analysis

In this section, we present a preliminary analysis of DynaShield's performance and cost.

Cost Analysis To estimate the cost of DynaShield, we estimate the cost of cloud services it builds on, and the profit it can gain from client solving cryptocurrency puzzles. We use AWS as our cost basis. We compute the cost of serverless functions in two cases: malicious bots solve the cryptocurrency puzzles vs. they do not solve the puzzles. When they do solve the puzzles, the cost of serverless functions can be computed as:

$$\text{AWS\_Lambda} = \mu_{Lambda} \times (b + g \times d) \times f \times 2 \qquad (1)$$

where μ_{Lambda} is the per million invocation cost of a Lambda function; b denotes how many millions of bots an attack uses; g denotes the legitimate users incoming request rate in the unit of million requests per second; d denotes the average attack duration; and f represents the average number of attacks a protected server experiences monthly. Each client's initial request will invoke a serverless function twice: one for puzzle fetching and the other for submitting the puzzle solution, hereby the ×2 factor. We also assume the capability tokens a server returns are valid for the entire attack period.

When malicious bots do not solve the puzzles, they will be blacklisted after a threshold of c tries (by default $c = 3$). We estimate the cost for serverless functions in this case as:

$$\text{AWS\_Lambda} = \mu_{Lambda} \times b \times c \times f + \mu_{Lambda} \times g \times d \times 2 \times f \qquad (2)$$

We estimate the cost of renting VMs as follows:

$$\text{AWS\_EC2} = \mu_{VM} \times d \times f \qquad (3)$$

where μ_{VM} is the cost of renting a VM. We use the price of AWS EC2 VMs listed at [4] for our estimate.

We estimate the profit from coin mining by assuming each client has at least a hashrate of $239H/s$, which is half of the hash rate of a commodity PC with an Intel i7-4790 CPU. Such a PC only costs $338 nowadays. We choose Monero Coin [18] as the cryptocurrency because it is ASIC-resistant. We obtain the profit of coin mining from an online cryptocurrency profit calculator [10]. The current price of Monero at the time of writing is $49.39. We assume the puzzle difficulty requires 5 seconds of each client's computation time.

As a comparison, we also estimate the cost of using Cloudflare for DDoS protection. We do not include the base fee for Cloudflare's estimate, because it differs by the type of clients. We compute Cloudflare's cost as

$$\mu_{Cloudflare} \times G_{Cloudflare} \qquad (4)$$

according to the price plan of Cloudflare [19], where $\mu_{Cloudflare}$ is the cost per million good requests destined to a protected server and $G_{Cloudflare}$ is the total number of good requests in millions. This calculation underestimates Cloudflare's cost.

We obtain the website traffic statistics from Alexa [12], and the attack traffic statistics from Incapsula [15] to estimate how much it may cost a server to use DynaShield and Cloudflare. We use `www.duke.edu` as an example. From Alexa's statistics, the number of Duke server's pageviews in December 2018 is $37,836,806$. We use this number as the number of good requests for Cloudflare's cost estimate. The num-

	Cost (-)/Profit (+) per month
AWS EC2: `t2.2xlarge`	-$0.897
AWS Lambda (bots solve puzzles)	-$1.624
AWS Lambda (bots do not solve)	-$2.436
Cloudflare	-$199.5
Monero(bots solve puzzles)	+$13.8
Monero(bots do not solve)	+$0.238

Table 1: **This table shows DynaShield's cloud service charges, cryptocurrency profit, and the cost of using Cloudflare for DDoS protection for a sample server.**

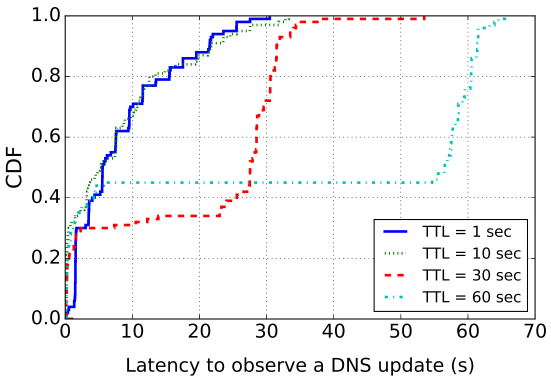

Figure 2: **This figure shows the latency distribution when a client observes a DNS record update after the update is made for varying TTL values of the DNS record.**

ber of hourly visits the Duke server had in that month is $10779/hr(2.99/sec)$, which we use as the legitimate clients' incoming request rate g. From Incapsula's statistics, the average number of DDoS attacks a target server experiences per month is 2.9 and the average duration is 50 minutes. We assume attack traffic comes from as many as 1 million bots. Thus, we set b to 1 million.

The cost of each invocation of Lambda is $0.28, where the flat cost is $0.2, and $.08 is the cost assuming a 5MB-Second usage. We obtain the memory usage number using a prototype implementation of the puzzle distribution and verification function. We choose a `t2.2xlarge` VM instance with 16 vCPUs and 32GB memory. Its cost is $0.3712 per hour. We choose this VM instance because it has sufficient compute resources to verify and forward all authorized requests based on the Duke server's traffic statistics.

With these assumptions, Table 1 shows the monthly cryptocurrency mining gain, the monthly cloud charges for DynaShield, and the monthly cost if the server uses Cloudflare.

The results show DynaShield costs as low as $\$0.897 + \$2.436 - \$0.238 = \$3.095$, even if we assume bots do not solve puzzles. This is only 1.6% of Cloudflare's cost, even if we discount the flat monthly fee of Cloudflare.

DNS Update Latency We evaluate how long it takes for a dynamic DNS service to replace the IP address of a protected server with those of DynaShield. In our experiment, we deploy AWS Route53 as the DNS service and set the DNS record TTL to different values. We then update the DNS record, and query the DNS name from 20 PlanetLab [26] nodes. We measure the time it takes for all vantage points to see the update after we make the update.

Figure 2 shows the results. When the DNS server uses a TTL less than 10 seconds, about 97% of the time all vantage points see the DNS update within 30 seconds. When the TTL is 30 seconds, about 98% of the time all vantage points see the update within 35 seconds. This result suggests that a server can direct the traffic to DynaShield quickly after it updates its DNS record.

Traffic Redirection Latency We evaluate the latency inflation caused by directing a client's traffic to DynaShield. We assume DynaShield may run on any of the three main cloud providers: AWS, Google Cloud Platform, and Microsoft Azure. Different cloud providers have sites at different geographical locations. We also assume DynaShield may run on different sites of each cloud provider. We choose 20 Planetlab nodes, a home computer, a Duke campus computer, and a node hosted by cPanel [9] as client and server nodes. For each pair of client and server nodes, we compare the latency when a client is redirected to DynaShield with the latency of a direct connection between a server and a client. We also measure the latency when a client's traffic uses Cloudflare to reach a server. In each experiment, we measure the latency for a client to fetch an empty file from a server. We use an empty file to discount the network transmission latency. We repeat each experiment 1000 times.

Direct	DynaShield	Cloudflare
100%	122%	118%

Table 2: **The average latency inflation compared to a direct connection's latency between a server and a client.**

Table 2 shows the latency inflation averaged among all client/server pairs and all seven different sites from three different providers. As can be seen, DynaShield's latency inflation is similar to that of Cloudflare and is 22% more than the direct connection latency on average.

5 Conclusion

This paper describes DynaShield, a low-cost DDoS solution for small organizations and individual Internet users. DynaShield is an application-layer capability-based architecture. It uses FaaS to process clients' initial requests, and uses IaaS to validate and forward clients' subsequent requests to a protected server. A server invokes DynaShield only when it is under attack, thereby avoiding cloud service charges during peacetime. DynaShield also uses cryptocurrency puzzles as Proof-of-Work to rate limit clients' initial requests. The profit from cryptocurrency mining can further reduce a server's cloud service cost. Our preliminary evaluation suggests that DynaShield can cost as little as a few dollars per month to prevent an organization from DDoS attacks.

6 Discussion

In this section, we discuss DynaShield's limitations as well as future work.

DynaShield uses cryptocurrency as proof-of-work to rate-limit clients' initial requests. Legitimate clients are also required to solve cryptocurrency puzzles, which may or may not be compliant with a server's policy. In addition, mobile clients may have limited computation resources. There also exist clients which do not enable JavaScript. To address these limitations, we can design DynaShield to allow a server to choose from a range of mechanisms such as proof-of-wait used in [28, 35] to rate limit clients' initial requests.

It is our future work to conduct a comprehensive evaluation on how DynaShield performs under various types of DDoS attacks. Presently, we focus on how we can reduce the cost of DynaShield. We have yet to study how DynaShield can adjust puzzle difficulties to match a server's bandwidth and processing capacity, and how to implement various admission policies set by a server.

The current DynaShield design only protects web services. It has the drawback that DynaShield must share a server's SSL/TLS private key to forward an HTTPs request to the server, similar to what a content distribution network does today. It is possible to extend DynaShield to protect other services by designing it to be a network layer or transport layer proxy, intercepting clients' traffic at the network or transport layer. It is our future work to investigate this approach.

It is also our future work to investigate when a server should stop using DynaShield. This problem is complicated because it depends on an attacker's attack strategy. If the attacker's attack strategy does not evolve, then a server can stop using DynaShield when there is no attack traffic reaching DynaShield. However, once a server deploys DynaShield, an attacker can evolve its attack strategy correspondingly. A strategic attacker can stop attacking a server right after it triggers DynaShield and start to attack the server again once it stops using DynaShield. However, launching such an on-off attack requires an attacker to have control over the attack infrastructure whenever the server stops using DynaShield. This will increase the attacker's cost for launching an effective attack. Under this attack strategy, a server may need to invoke and stop DynaShield multiple times to force the attacker to either expose its infrastructure or exhaust its attack fund. We plan to investigate the best strategy for the server to defend against this type of on-off attack in the future.

Acknowledgments

We thank the anonymous reviewers for their insightful comments, and this work was supported by an NSF award CNS-1345283.

References

[1] A Cisco Guide to Defending Against Distributed Denial of Service Attacks. https://www.cisco.com/c/en/us/about/security-center/guide-ddos-defense.html.

[2] A New Botnet Recently Started Recruiting IoT Devices. https://blog.radware.com/security/2018/02/jenx-los-calvos-de-san-calvicie/.

[3] Akamai DDoS Protection. https://www.akamai.com/uk/en/resources/ddos-protection.jsp.

[4] AWS EC2 Pricing. https://aws.amazon.com/ec2/pricing/on-demand/. Accessed on 05.08.2019.

[5] AWS Lambda vs EC2: Comparison of AWS Compute Resources. https://www.simform.com/aws-lambda-vs-ec2/.

[6] AWS Shield. https://aws.amazon.com/shield/.

[7] Azure DDoS Protection. https://azure.microsoft.com/en-us/services/ddos-protection/.

[8] Cloudflare: Protect Against DDoS Attack. https://www.cloudflare.com/ddos/.

[9] cPanel. https://www.cpanel.net/.

[10] CryptoCompare. https://www.cryptocompare.com/mining/calculator/xmr.

[11] DDoS Attacks in Q2 2018. https://securelist.com/ddos-report-in-q2-2018/86537/.

[12] Find Website Traffic, Statistics, and Analytics. https://www.alexa.com/siteinfo.

[13] Flow-Based Monitoring. https://www.akamai.com/us/en/products/security/flow-based-monitoring.jsp.

[14] Google Project Shield. https://projectshield.withgoogle.com/public/.

[15] Incapsula: Global DDoS Threat Landscape. https://www.incapsula.com/ddos-report/ddos-report-q4-2017.html.

[16] Mirai (malware). https://en.wikipedia.org/wiki/Mirai_(malware).

[17] Mitigate Volumetric DDoS Attacks with Nokia FP4 Silicon. https://pages.nokia.com/T001Z6.DDoS.attack.mitigation.lp.html.

[18] Monero (cryptocurrency). https://en.wikipedia.org/wiki/Monero_(cryptocurrency).

[19] Our Plans | Cloudflare. https://www.cloudflare.com/plans/.

[20] State of the Internet. https://www.akamai.com/es/es/multimedia/documents/case-study/spring-2018-state-of-the-internet-security-report.pdf.

[21] Universities Beware There's a New DDoS Attack Method. https://edtechmagazine.com/higher/article/2017/05/universities-beware-theres-new-ddos-attack-method.

[22] David G. Andersen. Mayday: Distributed Filtering for Internet Services. In *USENIX Symposium on Internet Technologies and Systems*, 2003.

[23] Tom Anderson, Timothy Roscoe, and David Wetherall. Preventing Internet Denial-of-Service with Capabilities. In *ACM HotNets*, 2003.

[24] Katerina Argyraki and David Cheriton. Network Capabilities: The Good, the Bad and the Ugly. In *ACM HotNets*, 2005.

[25] Katerina J Argyraki and David R Cheriton. Active Internet Traffic Filtering: Real-Time Response to Denial-of-Service Attacks. In *USENIX Annual Technical Conference*, 2005.

[26] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. Planetlab: An overlay testbed for broad-coverage services. *ACM SIGCOMM CCR*, 33(3), July 2003.

[27] Dixon Colin, Anderson Thomas E, and Krishnamurthy Arvind. Phalanx: Withstanding Multimillion-Node Botnets. In *ACM/USENIX NSDI*, 2008.

[28] Jon Crowcroft, Tim Deegan, Christian Kreibich, Richard Mortier, and Nicholas Weaver. Lazy Susan: Dumb Waiting as Proof of Work. Technical Report UCAM-CL-TR-703, University of Cambridge, Computer Laboratory, 2007.

[29] Dino Farinacci, Tony Li, Stan Hanks, David Meyer, and Paul Traina. Generic Routing Encapsulation (GRE). RFC 2784, 2000.

[30] John Ioannidis and Steven M Bellovin. Pushback: Router-based Defense against DDoS Attacks. In *NDSS*, 2002.

[31] Min Suk Kang, Virgil D Gligor, Vyas Sekar, et al. SPIFFY: Inducing Cost-Detectability Tradeoffs for Persistent Link-Flooding Attacks. In *NDSS*, 2016.

[32] Fayaz Seyed Kaveh, Tobioka Yoshiaki, Sekar Vyas, and Bailey Michael. Bohatei: Flexible and Elastic DDoS Defense. In *USENIX Security Symposium*, 2015.

[33] Angelos D Keromytis, Vishal Misra, and Dan Rubenstein. SOS: Secure Overlay Services. In *ACM SIGCOMM*, 2002.

[34] Xin Liu, Xiaowei Yang, and Yanbin Lu. To filter or to Authorize: Network-layer DoS Defense against Multimillion-Node Botnets. In *ACM SIGCOMM*, 2008.

[35] Xin Liu, Xiaowei Yang, and Yong Xia. NetFence: Preventing Internet Denial of Service from Inside Out. In *ACM SIGCOMM*, 2010.

[36] R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. Controlling High Bandwidth Aggregates in the Network. *ACM SIGCOMM CCR*, 32(3), 2002.

[37] Pedro Marques, Robert Raszuk, Danny McPherson, Jared Mauch, Barry Greene, and Nischal Sheth. Dissemination of Flow Specification Rules. RFC 5575, 2009.

[38] Bryan Parno, Dan Wendlandt, Elaine Shi, Adrian Perrig, Bruce Maggs, and Yih-Chun Hu. Portcullis: Protecting Connection Setup from Denial-of-Capability Attacks. In *ACM SIGCOMM*, 2007.

[39] Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson. Practical Network Support for IP Traceback. In *ACM SIGCOMM*, 2000.

[40] Elaine Shi, Ion Stoica, David Andersen, and Adrian Perrig. OverDoSe: A Generic DDoS Protection Service Using an Overlay Network. Technical Report CMU-CS-06-114, Carnegie Mellon University, 2006.

[41] Seung Won Shin, Phillip Porras, Vinod Yegneswara, Martin Fong, Guofei Gu, and Mabry Tyson. Fresco: Modular Compassable Security Services for Software-Defined Networks. In *NDSS*, 2013.

[42] Dawn Xiaodong Song and Adrian Perrig. Advanced and Authenticated Marking Schemes for IP Traceback. In *IEEE INFOCOM*, 2001.

[43] Abraham Yaar, Adrian Perrig, and Dawn Song. SIFF: A Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks. In *IEEE Symposium on Security and Privacy*, 2004.

[44] Xiaowei Yang, David Wetherall, and Thomas Anderson. TVA: a DoS-limiting Network Architecture. *IEEE/ACM Transactions on Networking*, 16(6):1267–1280, 2008.

[45] Liu Zhuotao, Jin Hao, Hu Yih-Chun, and Bailey Michael. MiddlePolice: Toward Enforcing Destination-defined Policies in the Middle of the Internet. In *ACM CCS*, 2016.