CDN Cannon: Exploiting CDN Back-to-Origin Strategies for Amplification Attacks

Ziyu Lin[1,3], *Zhiwei Lin*[2,3], *Ximeng Liu* [1✉], *Jianjun Chen*[3✉],
Run Guo[3], *Cheng Chen*[1], *Shaodong Xiao*[1]
[1]*Fuzhou University* [2]*Sichuan University* [3]*Tsinghua University*

Abstract

Content Delivery Networks (CDNs) provide high availability, speed up content delivery, and safeguard against DDoS attacks for their hosting websites. To achieve the aforementioned objectives, CDN designs several `back-to-origin` strategies that proactively pre-pull resources and modify HTTP requests and responses. However, our research reveals that these `back-to-origin` strategies prioritize performance over security, which can lead to excessive consumption of the website's bandwidth.

We have proposed a new class of amplification attacks called **Back-to-Origin Amplification (`BtOAmp`)** Attacks. These attacks allow malicious attackers to exploit the `back-to-origin` strategies, triggering the CDN to greedily demand more-than-necessary resources from websites, which finally blows the websites. We evaluated the feasibility and real-world impacts of `BtOAmp` attacks on fourteen popular CDNs. With real-world threat evaluation, our attack threatens all mainstream websites hosted on CDNs. We responsibly disclosed the details of our attack to the affected CDN vendors and proposed possible mitigation solutions.

1 Introduction

Content Delivery Networks (CDNs) play a pivotal role as intermediaries between global clients and website origins. The CDN's primary function is resource caching, a mechanism employed judiciously to optimize content delivery. CDN deploys millions of servers on the Internet backbone to offer services to global websites, working as a critical Internet infrastructure with massive computational, cache, and network resources [1]. Based on the available resources, to seek faster content delivery, CDN designs its `back-to-origin` strategies to arbitrarily and greedily demand more than necessary resources from the website origin (origin server) [2–4].

In this paper, we systematically examine `back-to-origin` strategies and present a novel class of amplification attacks called **Back-to-Origin Amplification (`BtOAmp`)** attacks, which allow attackers to exploit the vulnerabilities of CDN's `back-to-origin` strategies to launch an amplification attack against origin servers. We further classify the attacks into the following four kinds:

`Image Optimization` Attack. We exploit the CDN image optimization feature to launch a maximum 39039-factor amplification attack. It allows an attacker to abuse cropping images to one-pixel images, which can be used to launch an amplification attack against the origin server. We further analyzed the image optimization process and discovered that combining compression and cropping can further increase the amplification factor.

`Request Modification` Attack. We abuse CDN's `back-to-origin` configuration ability to amplify requests to the website origins, with a maximum amplification factor of 93077. CDN's `back-to-origin` strategies encompass the ability to manipulate HTTP headers and URLs. In our study, we analyzed fourteen CDNs and found that eleven of them allow the addition of large HTTP headers and the modification of a thin URL (e.g., "/a") into a much fatter URL (e.g., "/a" * 1024). Attackers can exploit it to exhaust the origin server's bandwidth. Even worse, attackers can cascade two or more CDNs together, leveraging multiple CDNs to achieve a better amplification factor.

`Method Conversion` Attack. We exploit CDN's request method conversion behavior to reach a maximum 53352-factor amplification attack. Through experiments, we discovered that CDNs convert HEAD requests into GET requests. This behavior allows attackers to launch an amplification attack against the origin server by exploiting HEAD-to-GET conversions. Specifically, the attacker sends HEAD requests, triggering the CDN to convert them into GET requests. Consequently, CDN requests all the complete resources from the origin server while returning only HTTP headers to the attacker.

`Connection Decoupling` Attack. As CDN maintains connections with the origin servers even after the client forcefully terminates the TCP connection [5], we exploit this behavior to launch an amplification attack with a 104862 factor.

✉Corresponding authors: snbnix@gmail.com, jianjun@tsinghua.edu.cn

In our research, we came across a few CDNs that terminate the CDN-origin connections when there is a disruption in the client-CDN connections. However, upon further analysis, we found that attackers can use `Transfer-Encoding: chunked` to force CDN to sustain CDN-origin connections, even after the attacker cuts off the client-CDN connections.

Contributions. We make the following contributions in this paper.

- *A Novel Attack.* We have proposed a new class of HTTP amplification attacks, Back-to-Origin Amplification (`BtOAmp`) attacks. These attacks exploit the CDN `back-to-origin` strategies, consuming the victim's incoming and outgoing bandwidth. It leads to network unavailability as well as economic losses.
- *Real-world Evaluations.* We examine the `BtOAmp` attacks on 14 popular CDN vendors and evaluate the feasibility and severity of `BtOAmp` vulnerabilities. We find all examined CDNs are vulnerable to the `BtOAmp` attacks, and the amplification factor is up to 104862 times in some cases.
- *Mitigation and Responsible Disclosure.* We present approaches to mitigate the proposed attacks and responsibly report vulnerabilities to CDN vendors. Notably, five CDN vendors acknowledged the vulnerabilities and took our mitigation solutions to fix them.

Roadmap. We organize the rest of this paper as follows. Section 2 provides the background of CDN and `back-to-origin` strategies. In Section 3 we describe the details of the `BtOAmp` attacks and explore the amplification factors. We also evaluate the feasibility of the `BtOAmp` attacks in Section 4. We discuss mitigation solutions and our responsible disclosure in Section 5. Section 6 elaborates on the related works, including CDN security, and amplification attacks. We conclude in Section 7.

2 Background

2.1 CDN Overview

CDN is a network of server clusters distributed globally, which has become an essential component of the Internet infrastructure. It not only enhances the performance of customer websites but also provides security features such as DDoS protection mechanisms [6]. As shown in Figure 1, CDN acts as a man-in-the-middle between clients and origin servers to decouple traditional client-origin connections into two segments: the client-CDN and the CDN-origin connections.

When a user requests a resource, CDN will attempt to respond from its cache [7]. If the cache is missing, it forwards the request to the origin server to obtain the desired resource and caches the response for subsequent requests. This mechanism efficiently reduces user access latency while relieving the load pressure on the origin server. Besides, CDNs dynamically select edge nodes by load balancing [8, 9]. In a word,

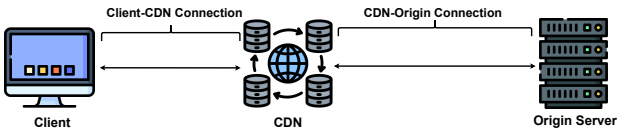


Figure 1: Multiple segments of connectivity in a CDN environment.

CDNs can not only reduce user access delay, but CDNs also serve as an effective DDoS defense for the origin server. To further improve cache hit rates, reduce user access latency, and cater to various user demands, CDNs continuously optimize their `back-to-origin` strategies.

2.2 Back-to-Origin Strategy

CDN `back-to-origin` strategies refer to the methods and rules employed by the CDN when it receives requests from clients and interacts with the origin server to retrieve the required source. The primary objective of the `back-to-origin` strategies is to increase the hit rate of CDN cache, effectively alleviate the load on the origin server, reduce network latency, and ultimately enhance the overall user browsing experience. `Back-to-origin` strategies are critical to CDN services. However, there are no related RFCs or commonly accepted industry standards. Consequently, `back-to-origin` strategies are implementation-dependent and may vary across different CDN vendors. When CDN vendors prioritize performance over security, threats may arise. To our knowledge, no research has systematically analyzed these strategies of various CDNs. In this study, we present the first differential testing of these strategies employed by 14 popular CDN vendors, and we identify five strategies that prioritize performance over security, as shown in Table 1. Additionally, to ensure compatibility with websites of various types, three out of these five strategies can be configured during the CDN service registration process.

Specifically, CDNs provide image optimization strategies that return images in different qualities and sizes based on client requirements. These strategies ensure the best balance of image loading speed and quality across various devices [10]. Additionally, CDNs offer Header Modification and URL Rewriting strategies that permit customers to modify back-to-origin requests and responses to meet specific needs. These strategies help to achieve more precise resource control. Moreover, to improve the CDN cache hit ratio, some CDNs will convert the HEAD requests to GET requests to retrieve more resources from the origin server [11]. Finally, some CDNs provide connection decoupling strategies in which CDNs may maintain the CDN-origin connection even after the client-CDN connection closes, allowing the CDN to continue caching requested resources. The first three strategies improve CDN performance, making it more flexible and adaptable to various user needs. The latter two strategies aim

Table 1: Back-to-Origin strategies.

	Alibaba	Azure	Baidu	Bunny	Cachefly	CDNetworks	ChinaNetCenter	Cloudflare	CloudFront	Edgio	Fastly	G-core	Qiniu	UPYun
Image Optimization†	✔			✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Header Modification†	✔	✔	✔	✔		✔	✔	✔	✔	✔	✔	✔		✔
URL Rewriting†	✔	✔	✔	✔		✔	✔	✔	✔	✔	✔	✔		✔
Method Conversion‡	✔			✔	✔	✔	✔	✔			✔	✔		
Decouple Connection‡	✔	✔	✔	✔	✔	✔	✔	✔				✔		✔

† These strategies are configurable.
‡ These strategies are the default strategies.

to increase the cache hit rate, optimizing response time when users subsequently access the same resources.

3 Back-to-Origin Amplification Attacks

To seek service performance and availability, CDNs commonly design and implement `back-to-origin` strategies to retrieve significantly more resources from the origin server than the client requests. However, we find that these strategies result in significant disparities in traffic between the client-CDN side and the CDN-origin side, which leads to an amplification attack.

In this section, we will introduce the threat model of the `BtOAmp` attack. We will then explain how to bypass the CDN cache and exploit these strategies to launch a `BtOAmp` attack. Finally, we will evaluate our techniques on fourteen leading CDN vendors to demonstrate the feasibility of amplification attacks based on different `back-to-origin` strategies.

We selected these fourteen CDN vendors based on CDN market share statistics [12] and from a Google search. Additionally, these fourteen CDN vendors are often studied in previous works as well [13–15]. We found that over 36.5% of websites on the Tranco Top 10,000 list [16] are hosted on these fourteen vendors. As the famous CDN vendors are all affected by the `BtOAmp` attacks, we believe our attacks are also applicable to other CDN vendors.

3.1 Threat Model

When a CDN applies the performance-over-security strategies to demand more than necessary resources from the origin server, we present that it leads to the `BtOAmp` attack. In this study, we assume that attackers possess two limited abilities. First, attackers can send malicious but legal requests to the CDN using a Cloud VPS worldwide or any hacked vantage point. Second, attackers can register accounts with CDN vendors, alter the CDN-offered configuration related to `back-to-origin` strategies, and host the victim website as the origin of the CDN service. Specifically, most CDN vendors currently offer free or free-trial services to potential website customers (including attackers) without robust authentication requirements [17, 18], possibly due to competitive reasons. It makes anonymous assaults feasible at a low cost or even free.

Within this context, grounded in differential testing principles, we systematically examine `back-to-origin` strategies and related configurations, aiming to uncover inconsistencies between the client-CDN side and the CDN-origin side. By leveraging these discrepancies, we delve into how attackers can craft legal requests to trigger CDN behaviors that demand more-than-necessary resources from the origin server (the victim website). Specifically, we investigate these inconsistencies at both the transport and application layers, assessing their amplification effects at the network layer. This analysis reveals that such legal but malicious requests may exhaust the origin server's limited bandwidth, leading to a `BtOAmp` attack on the origin server, as demonstrated in Figure 2.

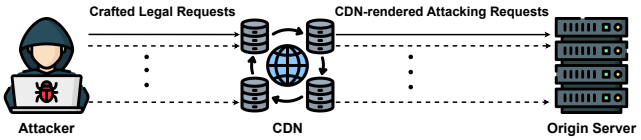


Figure 2: BtOAmp attack against CDN hosted websites.

With comprehensive case analysis, we identify and categorize the `BtOAmp` attack into four types according to the specific `back-to-origin` strategies: `Image Optimization` attack, `Request Modification` attack, `Method Conversion` attack, and `Connection Decoupling` attack, and the detailed attacking procedures are explained in the following subsections. Above all, we find that these fourteen CDN vendors all are vulnerable to the `BtOAmp` attack, and the details are summarized in Table 2.

3.2 Bypassing CDN Cache

To send successive attack requests to the victim, the presence of a CDN cache may prevent forwarding of attack requests to the victim. When attacking a website, the victim may return a cachable HTTP response. It is necessary to detour the CDN cache mechanism to ensure that attacking requests reach the victim rather than hitting the CDN cache.

After exploring the CDN forwarding strategies and working mechanism exhaustively, we conclude several approaches to bypass the CDN cache mechanism and confirm them in the 14 CDN vendors, as listed in Table 3.

CDN vendors cache resources based on HTTP-related parameters such as file extensions, URL paths, or HTTP Header. Attackers can exploit these rules to bypass CDN caching. Besides, we also found that some CDN providers forward

Table 2: CDN vendors vulnerable to the BtOAmp attacks.

	Alibaba	Azure	Baidu	Bunny	Cachefly†	CDNetworks‡	ChinaNetCenter	Cloudflare	CloudFront	Edgio	Fastly	G-core	Qiniu	UPYun
Image Optimization	✔			✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Request Modification*	✔	✔	✔	✔		✔	✔	✔	✔	✔	✔	✔		✔
Method Conversion	✔			✔	✔	✔	✔	✔			✔	✔		
Connection Decoupling	✔	✔	✔	✔	✔	✔	✔	✔				✔		✔

† Cachefly only provides image optimization services to enterprise customers.
‡ CDNetworks also only provides image optimization services to enterprise customers. However, we validated the feasibility of the image optimization attack with test samples from the official websites.
* Request Modification attack needs to register CDN account.

Table 3: Techniques to bypass CDN cache mechanism.

	Alibaba	Azure	Baidu	Bunny	Cachefly	CDNetworks	ChinaNetCenter	Cloudflare	CloudFront	Edgio	Fastly	G-core	Qiniu	UPYun
Dynamic Resouces	✔	✔	✔	✔		✔	✔	✔	✔	✔	✔		✔	✔
Random URL†	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Query Parameters	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
HTTP POST/PUT†	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
WebSocket handshake	✔	✔	✔	✔				✔	✔	✔	✔		✔	✔
Cookie Header	✔		✔	✔				✔		✔	✔		✔	✔
Authorization Header	✔	✔	✔	✔	✔			✔		✔	✔		✔	✔

† Random URLs and HTTP POST/PUT techniques can only apply to the request modification attack, and the other techniques can apply to all attacks.

requests with a Cookie or Authorization header to the victim and do not cache their responses.

3.3 Image Optimization Attack

In this section, we introduce a new class of amplification attacks called `Image Optimization` attacks. By exploiting the CDN image optimization features, attackers can create a substantial bandwidth disparity between the optimized client-side image delivery and the actual resource demanded from the origin server. This novel amplification attack targets the inherent functionalities within CDNs designed to enhance client-side image loading speeds, turning these optimizations into vectors for bandwidth exhaustion at the origin server.

3.3.1 Attack Surface

Primer on CDN image optimization. The image optimization feature of CDN is a service that processes and accelerates images at CDN nodes, improving client-side image loading speed and quality while saving bandwidth and costs. It offloads the website's burden to prepare an image with different resolutions to meet the needs of various clients with different screen sizes. Among these, image compression and image cropping are two common image optimization techniques [10, 19].

Image compression is one of the CDN image optimization features, aiming to help clients save bandwidth and costs while improving image loading speed. The fundamental principle of image compression is to reduce the file size of images with image format conversion or image quality adjustment thus improving the efficiency of image transmission [20].

Image cropping functionality is another CDN image optimization feature, aiding websites in adapting to various client-side screens and devices. The principle behind image cropping involves adjusting the image's sizes to accommodate different screens and devices [21]. The dimensions of an image, determined by its width and height, play a pivotal role in determining the overall file size. Larger image sizes result in larger file sizes and crisper display quality, while smaller image sizes reduce file size but may lead to a less clear display.

In summary, CDN's image optimization features assist websites in better managing and displaying images, thereby improving user experiences and website performance.

Attack Principle. The core of the `Image Optimization` attack lies in its ability to exploit the CDN's image optimization functionalities, specifically image compression, and cropping, to exert undue pressure on the origin server's bandwidth. By inducing the CDN to bypass its caching mechanisms for image optimization requests, attackers force the retrieval of lossless, high-resolution images from the origin server. Meanwhile, the CDN's subsequent compression and delivery of these images helps to reduce client-side (also attacker-side) bandwidth consumption, resulting in amplification attacks. This discrepancy is further exacerbated when attackers manipulate image cropping features, compelling the CDN to fetch and process large images only to minimal sizes for delivery, as shown in Figure 3. Such strategies not only inflate the bandwidth demand on the origin server but also underscore the vulnerabilities inherent in CDN image optimization services.

The `Image Optimization` attack demonstrates a clever abuse of CDN's intended optimization functionalities, highlighting the need for enhanced security awareness and measures to reduce potential vulnerabilities within these optimization processes.

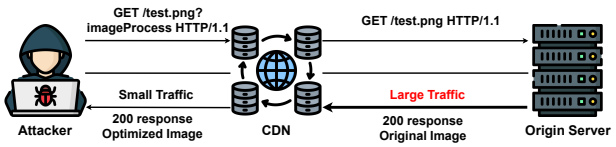


Figure 3: Concept of Image Optimization attack. Step 1: Exploiting CDN edge servers to request original images; Step 2: CDN optimizes images to reduce image size.

3.3.2 Amplification Factor

Image optimization attacks, including image compression and image cropping attacks, are evaluated for their amplification factors through two distinct experiments.

Common Experimental Setup. Both experiments were conducted with an Nginx web server deployed as the origin server across various CDNs. The traffic on both client-CDN and CDN-origin connections was captured using the `tcpdump` [22]. This setup serves as the foundation to assess the impact of image characteristics: the first experiment focused on image formats, while the second concentrated on image resolutions.

The Setup of the First Experiment. For the initial experiment, images in four common formats (PNG, JPG, BMP, and TIFF) were placed on the Nginx server. We sent requests requiring the CDN to fetch and compress these images into WebP format, evaluating the effect of image format on the amplification factor.

Results of the First Experiment. As shown in Table 4, the maximum amplification factor achieved through image compression attacks was 1011. An interesting behavior was observed with Cloudflare's handling of TIFF and BMP formats, where it would return a `415 Unsupported Media Type` response but still fetch the original image, presenting an opportunity for a more potent amplification attack.

Table 4: The amplification factor varies with the format of the image in the Image Optimization attack.

	PNG	JPG	BMP	TIFF
Alibaba†	111	80	126	N/A
Bunny†	136	98	N/A	N/A
ChinaNetCenter†	130	94	156	N/A
Cloudflare†	319	230	1011	1011
CloudFront‡	23	17	N/A	26
Edgio‡	23	17	N/A	26
Fastly†	1.7	1.2	N/A	N/A
G-core†	139	100	N/A	N/A
Qiniu‡	30	21	25	34
UPYun†	139	101	166	149

† These CDNs support lossy compression.
‡ These CDNs support lossless compression.

The Setup of the Second Experiment. The second experiment focused on image resolution. Images of various resolutions, from 720p to 4320p, were evaluated. Requests were crafted to require the CDN to crop these high-resolution images into one pixel, aiming to discern the resolution's impact on amplification.

Results of the Second Experiment. As shown in Figure 4, the amplification factor tended to correlate with the image resolution. However, Alibaba and Edgio were noted to support image optimization only up to a 4096x4096 resolution, directly delivering back the original image for resolutions above this threshold. Additionally, certain CDNs like Alibaba, Qiniu, UPYun, and ChinaNetCenter, when receiving requests with invalid optimization parameters, would issue a `400 Bad Request` response yet still forward the request to fetch the image, thereby still enabling an `BtOAmp` attack.

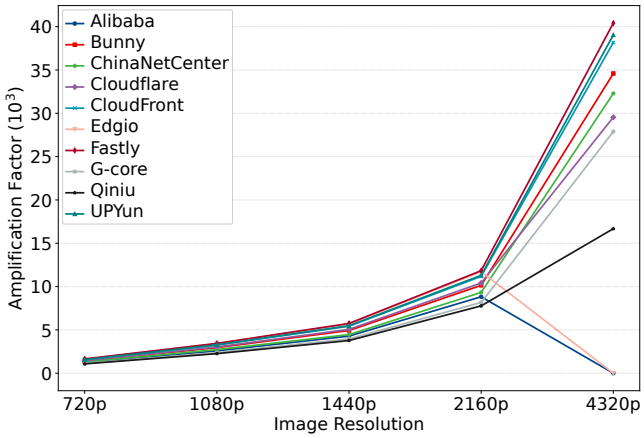


Figure 4: How the amplification factor changes with the resolution of images.

3.4 Request Modification Attack

In this section, we will delve into the Request Modification Attack. This attack leverages CDN infrastructure, which allows for the amplification of attacks by adding HTTP headers and modifying URLs to requests. In contrast to traditional DDoS attacks that rely on massive bots [23–25], this attack is stealthier and harder for CDN to defend against because the crafted requests are legal and are initiated from itself.

3.4.1 Attack Surface

Primer on CDN Modification HTTP Header Behaviors. Chen et al. [17] found that CDNs typically add default headers such as Via and X-Forwarded-For headers when forwarding requests and exploit these automatically added headers to launch amplification attacks against CDNs. However, we found that CDN-offered configurations allow website customers (also the attacker) to add a large number of huge HTTP Headers and rewrite a thin URL (e.g., "/a") into a much fatter URL (e.g., "/a" * 1024) in forwarding requests. Therefore, we present a new class of amplification attacks that attackers can exploit to launch amplification attacks against origin servers.

Table 5: The amplification factor in Request Modification attack.

	Alibaba	Azure	Baidu	Bunny	CDNetworks	ChinaNetCenter	Cloudflare	CloudFront	Edgio	Fastly	G-core	UPYun
Header Name Size	256B	128B	128B	≥1MB	64B	64B	128B	128B	≥100KB	255B	255B	40B
Header Value Size	256B	640B	1000B	≥1MB	63B	64B	512B	768B	≥100KB	≥10KB	512B	400B
Number of Headers	49	99	20	≥10	≥1300	≥800	270	10	15	≥13	49	20
URL Size	≥50KB	512B	1000B	≥50KB	1KB	≥1KB	8KB	256B	10KB	N/A	N/A	400B
Host Header Size	≥512B	128B	64B	≥64B	64B	≥54B	N/A	N/A	≥128B	255B	2048B	≥128B
Amplification Factor	348	367	109	93077	768	481	846	43	5352	590	188	42

The `back-to-origin` strategies empower websites (also the attacker as a malicious CDN customer) with the capability to incorporate bespoke HTTP headers into forwarding requests, thereby catering to specific requirements and affording a higher degree of precision in resource management. When employing CDN to forward client requests, this CDN-provided request modification feature helps websites add various HTTP header fields, such as customized authorization tokens, access controls, and security tokens [26]. As a result, websites can exert a heightened and finely-tuned degree of control over the generated responses. Essentially, this flexibility empowers attackers to finely tune and personalize interactions with the origin server, optimizing the overall content delivery process for improved performance and functionality. However, it is essential to note that CDNs have lenient restrictions on the size and number of HTTP headers when adding them to forwarding requests, as shown in Table 5. As a result, attackers could take advantage of this vulnerability by leveraging CDNs to send requests with large HTTP headers, exhausting the origin server's bandwidth.

Primer on CDN Manipulation of URL Behaviors. It is common for a website to change its resource path on the server, while it requires the website to synchronize all the related URLs in the web pages. To avoid this annoying work, CDNs offer support for URL rewriting when forwarding client requests [27]. It is worth noting that this rewriting does not impact the CDN's internal routing or cache key. Nevertheless, it is crucial to acknowledge that CDNs tend to impose relatively permissive limitations on the length of modified URLs, as shown in Table 5. An attacker can abuse these strategies to make CDNs send requests carrying large URLs, leading to amplification attacks.

Primer on Cascadable CDN Platforms. Chen et al. [17] presented that two or more CDN platforms can be cascaded to make a request forwarding chain, and this cascadable CDN trick has been applied in `RangeAmp Attack` [28] to increase the damage of the attack. A necessary condition for cascading two or more CDN platforms in a chain is that the front-level CDN must forward the request to the next-level CDN, and the next-level CDN treats it as a benign request and continues forwarding it.

Whether the next-level CDN accepts the forwarded requests depends on the Host header. If the front-level CDN platform can modify the Host header in the forwarded requests to match the domain name registered for service on the next-level CDN platform, then the requests can be processed and forwarded by the next-level CDN platform. Our measurements show that thirteen out of fourteen CDN vendors can be cascaded at any level of the CDN chain, as shown in Table 6. However, Cloudflare does not support modifying the Host header as the origin domain name. We can only cascade Cloudflare as the last level in a CDN chain.

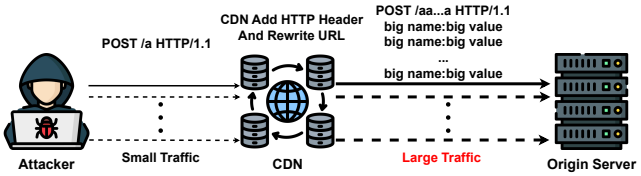


Figure 5: Concept of Request Modification attack. Step 1: Send numerous POST requests to CDN; Step 2: Exploiting CDN to make POST requests to "fat" POST requests and forward them to the origin server.

Attack Principle. With our extensive measurements, CDN `back-to-origin` strategies support the addition of HTTP headers and URL modifications within forwarding requests (Table 1). Hence, we present that attackers can exploit the CDN-provided request modification capability to launch the `Request Modification` attack. As shown in Figure 5, an attacker sends POST requests to the CDN first, then triggers the CDN to append numerous large HTTP headers and modify the URL into a fat URL, creating abnormally large requests that lead to amplification attacks.

Besides, we propose a variant of the `Request Modification` attack called the `Cascade Request Modification` attack. This attack exploits the cascading nature of CDNs to amplify the damage of the `Request Modification` attack. It allows attackers to exploit the cascaded CDNs. Each CDN contributes to the enlargement of requests by adding extensive HTTP headers, increasing the request size at each stage. We will delve into a detailed assessment of the cascadable CDN trick and real-world evaluation of the `Cascade Request Modification` attack in Section 4.

3.4.2 Amplification Factor

The size of the HTTP headers and URL emerges as a pivotal technical factor influencing the amplification factor of

Table 6: Modifiability of "Host" header field on CDNs.

	Alibaba	Azure	Baidu	Bunny	Cachefly	CDNetworks	ChinaNetCenter	Cloudflare	CloudFront	Edgio	Fastly	G-core	Qiniu	UPYun
Request Domain	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔
Origin Domain	✔	✔	✔	✔	✔	✔	✔	N/A (Free Plan)	✔	✔	✔	✔	✔	✔
Any Domain	✔	✔	✔	✔		✔	✔	N/A (Free Plan)		✔	✔	✔	✔	✔

the attack. To thoroughly assess its impact, we devised the following experiment.

Experiment Setup. Based on the above measurements of the various CDN `back-to-origin` strategies, we configured the strategies to append the maximum HTTP header and modify the raw URL to its maximum permitted length in forwarding requests. Subsequently, we sent separate POST requests to each CDN, which triggered the CDN to transfer them into large-sized ones. We captured and analyzed all the requests on both the clients and origin servers to determine the amplification factors.

Experiment Result. The most noteworthy experiment result is that twelve CDN vendors are affected by the `Request Modification` attack, as shown in Table 5. Remarkably, CDNs such as Alibaba, Bunny, and Edgio, which permit the addition of substantial, large-sized HTTP headers, exhibited higher amplification factors.

3.5 Method Conversion Attack

Through comprehensive analysis across client-CDN and CDN-origin connections, we've identified a nuanced behavior among CDNs concerning the support and conversion of HEAD requests. Specifically, our investigation into fourteen CDNs revealed that seven exhibit a vulnerability that could be exploited to amplify bandwidth against the origin servers they host.

3.5.1 Attack Surface

Primer on HTTP HEAD Request. The HEAD request aims to only retrieve metadata or header information about a specific web resource [29]. Unlike GET requests, HEAD requests are designed to exclusively retrieve information about the web resource without fetching the actual content. The HEAD request empowers the client to acquire essential information concerning the web resource, encompassing attributes like content length, content type, response status code, and so on, without downloading the complete resource.

The distinctive characteristics of HEAD requests make them exceedingly advantageous for performance optimization, bandwidth management, and cache control [30]. By exclusively obtaining header information and not downloading the whole content, HEAD requests efficiently reduce unnecessary network traffic, improve operational efficiency, and support caching mechanisms, thereby alleviating server and network burdens.

Method Conversion Behavior. While most CDN vendors assert support for HEAD requests to save traffic, a prevalent operational practice involves converting these HEAD requests into GET requests. This conversion aims to prefetch content, improve cache hit rates [11], and minimize response times for subsequent requests. Despite the prevalence of this behavior, there has not been a comprehensive study of its security.

In this experimental setup, we initially configure the origin server to support HEAD requests. Subsequently, we send HEAD requests to the CDNs and capture CDN-origin requests at the origin server using `tcpdump` to explore the CDN's `back-to-origin` strategy of HEAD requests. As shown in Table 7, experiment results have revealed that CDNs support HEAD requests in the client-CDN connection but exclusively use GET requests in the CDN-origin connection, even when the origin server supports HEAD requests.

Table 7: Behavior of CDNs forwarding HEAD requests.

	HTTP Method	
	client-CDN Connection	**CDN-origin Connection**
Alibaba	HEAD	GET
Azure	HEAD	HEAD
Baidu	HEAD	HEAD
Bunny	HEAD	GET
Cachefly	HEAD	GET
CDNetworks	HEAD	GET
ChinaNetCenter	HEAD	GET
Cloudflare	HEAD	GET
CloudFront	HEAD	HEAD
Edgio	HEAD	HEAD
Fastly	HEAD	GET
G-core	HEAD	GET
Qiniu	HEAD	HEAD
UPYun	HEAD	HEAD

Attack Principle. Through the above experiments, we have identified two distinct `back-to-origin` strategies CDNs employ to handle HEAD requests. In one scenario, upon receiving HEAD requests from clients, the CDN directly forwards them to the origin server. In the alternative scenario, the CDN converts HEAD requests into GET requests, fetches more than the required target resources from the origin server, and caches them in the CDN global cache.

As shown in Figure 6, the method conversion strategy results in the CDN retrieving the target resource more than necessary from the origin server and caching it while solely returning HTTP headers to the client. Consequently, it creates an asymmetry in bandwidth consumption between CDN-origin and client-CDN connections, offering attackers an ap-

proach to exhaust the origin server's bandwidth. Upon a HEAD request of a specific resource, the CDN converts it to a GET request and then caches the response for the following HEAD or GET requests of the same resources. However, attackers can use the cache bypass techniques to make each HEAD request not hit the cache and be forwarded to the origin server, thus exhausting the origin server's bandwidth, even when the same resource is requested many times.

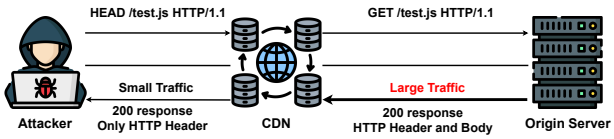


Figure 6: Concept of Method Conversion attack. Exploiting CDN edge servers to convert HEAD requests into GET requests, requesting resources from the origin server but responding with only HTTP headers.

3.5.2 Amplification Factor

We experimented to explore the practicality and amplification factor of the `Method Conversion` attack. This experiment aims to determine which CDNs are vulnerable and the resulting amplification factor.

Experiment Setup. We conduct our experiments under controlled conditions. We send a series of HEAD requests to CDNs to gather information regarding target resources on the origin server. For these experiments, we deployed a website on each CDN and placed target resources of varying sizes on the website origin server.

To determine the impact of the targeted resource size on the amplification factor, we systematically sent HEAD requests to the CDNs for targeted resources ranging from 1MB to 25MB, incrementing by 1MB. We used `tcpdump` to capture traffic on both the client-CDN and CDN-origin connections and calculate the amplification factor.

Experiment Result. As shown in Figure 7, the amplification factor is proportional to the size of the target resource for each CDN except Fastly. For every CDN, we summarize in Table 8 the amplification factor for a targeted resource of different sizes, i.e., 1MB, 10MB, and 25MB.

The `Method Conversion` attack can result in an amplification factor of hundreds of thousands by simply requesting a 10MB resource, which is a common occurrence on the internet. However, Fastly appears to be less affected by this attack. Upon further investigation, we discovered that although Fastly converts HEAD requests to GET requests, it immediately closes the CDN-origin connection after receiving the response header. This unique behavior prevents the amplification factor from increasing with the variations in the size of the target resource, which can act as a mitigation for the attack.

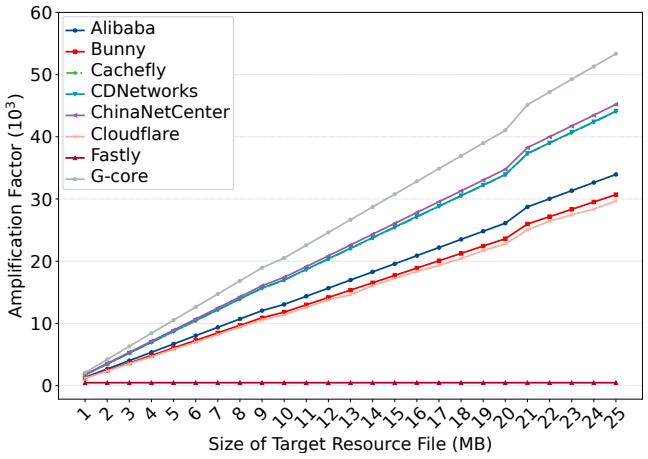


Figure 7: How does the Method Conversion attack amplification factor change with the size of a target resource file.

Table 8: Amplification factors with different target resource sizes of Method Conversion attacks.

	Amplification Factor		
	1MB	**10MB**	**25MB**
Alibaba	1340	13059	33952
Bunny	1212	11808	30702
Cachefly	1738	16940	44044
CDNetworks	1744	16995	44115
ChinaNetCenter	1784	17418	45212
Cloudflare	1170	11385	29698
Fastly	469	469	469
G-core	2106	20520	53352

3.6 Connection Decoupling Attack

It is well-known that CDN, working as a man-in-the-middle, decouples the traditional client-origin connection into two distinct connections: the client-CDN connection and the CDN-origin connection. When the client cuts off the client-CDN connection, no RFCs or industry standards define whether the specific CDN-origin connection should be kept open. In other words, this decoupling connection strategy remains implementation-specific, and some CDNs still maintain CDN-origin connections. Consequently, attackers can exploit this strategy to launch an amplification attack, exhausting the origin server's limited bandwidth.

The `Connection Decoupling` attack leverages the independence of client-CDN and CDN-origin connections. Attackers leverage CDN edge servers to download target resources from the origin server and prevent CDN from responding to these resources, thus consuming the bandwidth resources of the origin server.

3.6.1 Attack Surface

Traditional CDN decoupling attack is unusable. CDNs introduce a division in HTTP communication, delineating it into two discrete segments: the client-CDN connection and the CDN-origin connection. This strategy yields a variety of advantages, notably enhancing the performance and reliability of CDNs. Decoupling these connections and absorbing a substantial influx of flood traffic are common strategies for effectively countering DoS attacks. However, this strategy can give rise to security challenges stemming from potential inconsistencies in TCP states and the asymmetry in bandwidth between client-CDN and CDN-origin connections.

Triukose et al. [31] introduced a new type of DoS attack targeted at CDN-hosted website servers. In this attack, the attacker sends multiple requests directly to various ingress nodes, all requesting the same large file hosted on the website. These requests include a randomly generated query string attached to the URL to bypass the CDN cache. Subsequently, the attacker cuts off all client-CDN connections while the CDN continues to sustain CDN-origin connections, which operate at a significantly higher bandwidth. This results in a bandwidth exhaustion DoS attack against the website. However, during our experiments, we unearthed a noteworthy observation: certain CDNs automatically terminate CDN-origin connections once the Client-origin connection is severed, as shown in Table 9. This mitigation measure renders the traditional CDN decoupling attack outlined in [31] ineffective.

Table 9: Whether CDNs keep CDN-origin connection while an attacker cuts off client-CDN connection.

	CDN-origin Connection	
	Content-Length	**Transfer-Encoding**
Alibaba	Close	Keep
Azure	Close	Keep
Baidu	Close	Keep
Bunny	Keep	Keep
Cachefly	Keep	Keep
CDNetworks	Close	Keep
ChinaNetCenter	Keep	Keep
Cloudflare	Close	Keep
CloudFront	Close	Close
Edgio	Close	Close
Fastly	Close	Close
G-core	Keep	Keep
Qiniu	Close	Close
UPYun	Keep	Keep

Reviving the attack with Transfer-Encoding: Chunked. `Transfer-Encoding: Chunked` constitutes a data transmission technique within the HTTP protocol, wherein data is segmented into small chunks and transmitted sequentially to the client [32]. The principal advantage of this transmission technique lies in its capacity to generate and deliver data in real-time during the transmission process without waiting for the entire content to be ready [33]. This technique fosters not only flexibility but also substantial performance enhancements. For example, in online video conferencing, where real-time communication is vital, the ability to transmit and render data as it becomes available significantly reduces latency [34].

As shown in Table 9, it's important to note that certain CDNs automatically close CDN-origin connections when client-CDN connections are severed. However, we found that using `Transfer-Encoding: Chunked` can force the CDN to sustain CDN-origin connections even after the termination of client-CDN connections. It gives the attacker a new attack vector that revives the CDN decoupling attack. Attackers could still launch an amplification attack that exhausts the origin server's bandwidth.

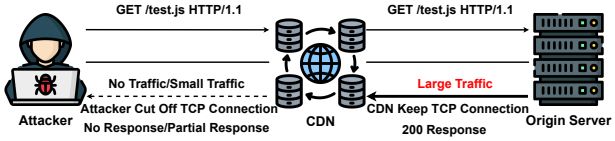


Figure 8: Concept of Connection Decoupling attack. Step 1: The attacker sends requests to CDN nodes; Step 2: The attacker closes the TCP connection and prevents the CDN from responding.

Attack Principle. When an attacker sends GET requests to a CDN and cuts off the attacker-CDN connections, the CDN forwards these requests to the origin server while still maintaining the CDN-origin connections, leading to amplification attacks. As shown in Figure 8, this strategy results in responses traveling only from the origin server to the CDN. It causes the bandwidth consumption on the CDN-origin connections to be significantly higher than the attacker-CDN connections.

3.6.2 Amplification Factor

We speculate that the size of the target resource influences the amplification factor of the `Connection Decoupling` attack. We conducted experiments to explore how the size of the target resource affects the amplification factor.

Experiment Setup. In our experiments, we deploy an Nginx web server on each CDN. We placed target resources of varying sizes on the origin server, ranging from 1MB to 25MB in increments of 1MB. We then used the `tcpdump` tool to capture traffic on client-CDN and CDN-origin connections to calculate the amplification factor. For each CDN, we first sent GET requests for different-sized target resources, then immediately cut off TCP connections to evaluate the impact of resource size on the amplification factor.

Experiment Result. As shown in Figure 9, the amplification factor for each CDN is essentially proportional to the size of the target resource. G-core is an exceptional case, as it differs from the prevalent implementation observed in most

other CDNs. In contrast to the standard behavior, wherein CDNs promptly close the TCP connection upon receipt of a client-sending RST frame requesting connection closure, G-core exhibits a distinctive implementation. This implementation entails a refusal to terminate the TCP connection until it has successfully filled the TCP window. However, it is worth noting that an attacker can still bypass this unique implementation in G-core by setting up a small TCP window to prevent CDN from responding.

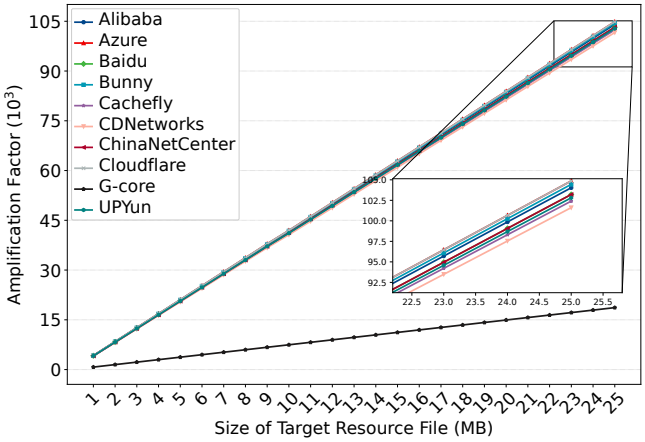


Figure 9: How does the Connection Decoupling attack amplification factor change with the size of a target resource file.

Table 10: Amplification factors with different target resource sizes of Connection Decoupling attacks.

	Amplification Factor		
	1MB	**10MB**	**25MB**
Alibaba	4181	41614	104029
Azure	4216	41948	104862
Baidu	4148	41286	103210
Bunny	4198	41779	104443
Cachefly	4116	40963	102403
CDNetworks	4082	40645	101609
ChinaNetCenter	4129	41284	103207
Cloudflare	4214	41946	104861
G-core	747	7469	18672
UPYun	4131	41124	102805

Table 10 summarizes the amplification factor for a target resource of different sizes (1MB, 10MB, and 25MB). The amplification factor is proportional to the target resource size.

4 Real-world Evaluation

When attack requests reach a website and simultaneously demand the utilization of internal bottleneck resources, such as CPU, memory, and logical queues, a significant influx of requests can persistently overwhelm these resources, leading to more severe DoS disruptions. We first evaluated the real-world amplification factor for all vulnerable CDNs. As shown in Figure 10, all CDNs we tested are vulnerable to at least one specific `BtOAmp` attack, and the `Image Optimization` attacks can attack ten CDN vendors, and it has the most extensive attack surface.

To further demonstrate the real-world impact of `BtOAmp` attacks on bandwidth amplification, we choose one different CDN for each specific `BtOAmp` attack to study the bandwidth of both the attacker and the origin server, results are shown in Figure 11(a) and Figure 11(b).

The Setup of Real-World BtOAmp attacks. We initially set up an HTTP service (2.5GHz/2GB/100Mbps/Nginx 1.21.3) in Silicon Valley as the target victim. Subsequently, we launched the `BtOAmp` attack in a VPS (2.5GHz/2GB/30Mbps) in Singapore. Both servers are located on the backbone network, which greatly reduces the risk of saturating on-path network segments with our attack traffic. To assess the impact of attacks, we monitor the bandwidth consumption of both the attacker (client) and the origin server.

Setting of Image Optimization Attack. Take Fastly as an example, we concurrently send three image optimization requests to Fastly every second, lasting 60 seconds. The resolution of the requested target image on the origin server is 4320 pixels.

Severity Analysis of Image Optimization Attack. Figure 11(a) and Figure 11(b) compare the bandwidth consumption. The attacker sacrifices a bandwidth consumption not exceeding 10Kbps, while the website's bandwidth achieves 100 Mbps at maximum.

Setting of Request Modification Attack. Taking Edgio as an example, we concurrently send 15 POST requests to Edgio per second for 60 seconds. Edgio appends 15 HTTP headers of 200KB each to these requests and rewrites the URL from "/a" to "/a" ten thousand times. Moreover, we also use CDN cascading technology to amplify the threat of `Request Modification` attacks. For example, we cascaded CloudFront and Bunny to evaluate the effectiveness of the `Cascade Request Modification` attack. We simultaneously send 3 POST requests to CloudFront every second for 60 seconds. CloudFront appends 10 HTTP headers, each 896B in size, to these requests and forwards them to Bunny. In addition, Bunny appends 10 HTTP headers of 2MB each to these requests and rewrites the URL from "/a" to "/a" a hundred thousand times.

Severity Analysis of Request Modification Attack. Figure 11(a) and Figure 11(b) demonstrate that an attacker can launch request modification attacks to bring down a victim's 100Mbps bandwidth with just an average 35kbps bandwidth. In addition, an attacker can also use CDN cascading techniques to launch `Cascade Request Modification` attacks, only needing to consume 15kbps bandwidth to paralyze the victim's 100Mbps bandwidth. CDN cascading techniques can greatly increase the threat of `Request Modification` attacks.

Setting of Method Conversion Attack. Taking Cloudflare

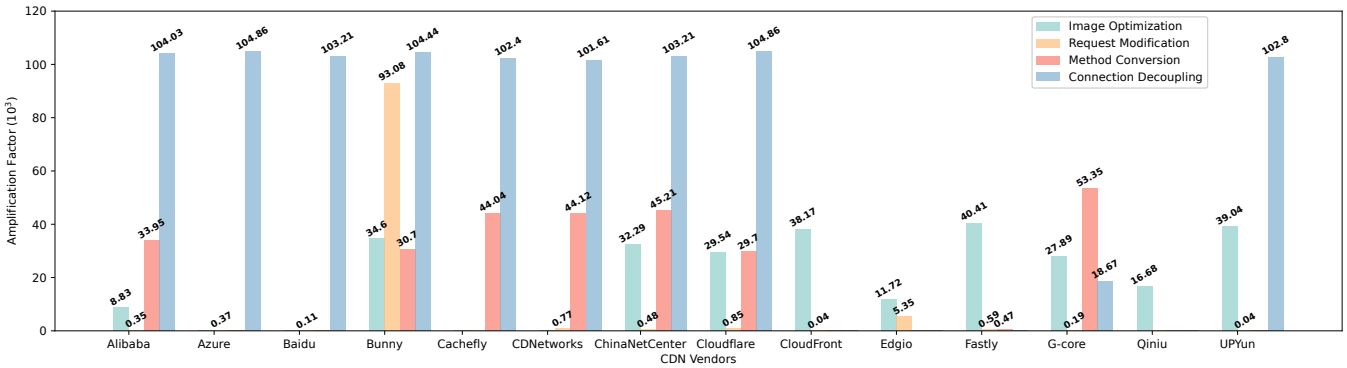
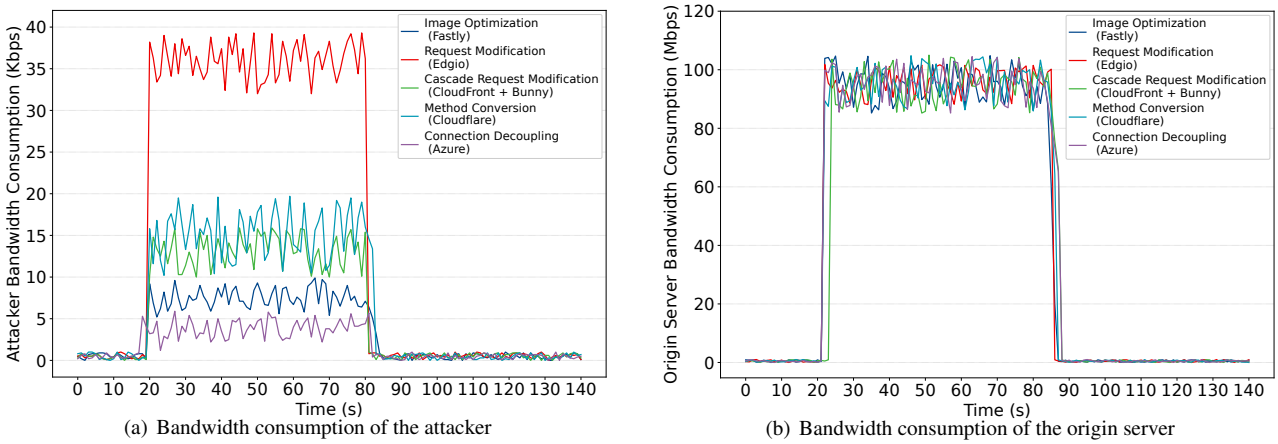


Figure 10: Amplification factor of the BtOAmp attacks on all CDN vendors.



(a) Bandwidth consumption of the attacker

(b) Bandwidth consumption of the origin server

Figure 11: The bandwidth consumption of the attacker and the origin server for BtOAmp attacks.

as an example, we concurrently send 2 HEAD requests per second to Cloudflare, lasting 60 seconds. The target resource size on the origin server is 10MB.

Severity Analysis of Method Conversion Attack. Figure 11(a) and Figure 11(b) show that an attacker can use below 10Kbps bandwidth to launch `Method Conversion` attacks to exhaust the victim's 100 Mbps bandwidth.

Setting of Decoupling Connection Attack. Taking Azure as an example, we concurrently send 2 GET requests to Azure every second, promptly cutting off the TCP connection upon sending the requests, lasting 60 seconds. The target resource size on the origin server is 10MB.

Severity Analysis of Decoupling Connection Attack. Figure 11(a) and Figure 11(b) show that an attacker can launch `Connection Decoupling` attacks by cutting off the TCP connection after the request has been sent. The attacker's bandwidth usage remains below 7Kbps, whereas the website's bandwidth reaches a maximum of 100 Mbps.

The above evaluations show that the `BtOAmp` attack ideally matches the "small spark, big fire" attacking philosophy. Worse, the stealthy nature of a `BtOAmp` attack can also help it to escape DDoS detection.

5 Discussion

5.1 Mitigations

In general, the presence of vulnerabilities is partly due to market competition. CDN vendors naturally aspire to offer more features and maximize compatibility with websites of varying configurations. However, CDNs serve as the backbone of the Internet. Therefore, if there are weaknesses in CDN implementations, they could potentially be exploited, jeopardizing the security of websites and causing a severe threat to the Internet.

Limit parameters in the Back-to-Origin strategies. Currently, all the CDN vendors we tested that support image optimization do not impose any restrictions on parameters. We recommend that CDNs set image cropping parameters to common resolutions in their default configurations, with users having the option to customize as needed. The CDN should cache the original image to reduce the workload on the origin server. Additionally, it is advisable to impose limitations on HTTP header size, HTTP header quantity, and URL size when implementing strategies for adding HTTP headers and rewriting URLs.

Follow RFC standards for request forwarding. We recommend that CDNs by default directly forward HEAD requests rather than automatically converting them into GET requests. If websites want to improve cache hit rates and speed up content loading, they can configure the CDN to forward HEAD requests as GET requests.

Synchronize client-CDN and CDN-origin connections. We recommend that CDNs wait several seconds after the client-CDN connection breaks, allowing the client to reconnect. If the client fails to reconnect, CDN should timely terminate the CDN-origin connection.

Enforce CDN to validate origin ownership. Currently, all fourteen CDNs we tested do not validate the ownership of customer-hosted origin, which allows attackers to register a CDN service and configure targeted websites as the origin [35, 36]. We recommend that CDN providers implement origin ownership validation to prevent abuse, which can narrow the scope of threats only on websites hosted on CDNs.

It's worth noting that both Alibaba and G-core have implemented our proposed solution to mitigate the `Image Optimization` attack. They did this by caching the original image on edge servers, which prevents the `Image Optimization` attack.

5.2 Ethic Consideration

During our study, we aimed to balance real-world severity evaluations with minimizing risks that could impact CDN vendors. We were concerned that higher bandwidth exhaustion during our tests could impair the CDN's network performance and harm other hosted websites. As a result, we took great care to avoid any ethical issues during our experiments.

Firstly, we strategically selected attacker positions with minimal latency to CDN nodes, ensuring our attacks are initiated within the backbone network. This configuration significantly reduced the risk of our attack traffic saturating on-path network segments. Secondly, our attacks targeted our own websites, keeping traffic below the CDN's capacity to avoid service disruption. Thirdly, we limited attack bandwidth to 100Mbps to prevent network overload and protect other CDN services. Fourthly, We only sent three requests to validate the feasibility of Cachefly's `Image Optimization` attack and that there would be no impact on the website's service. These approaches ensure our research minimizes harm while enhancing CDN security.

5.3 Responsible Disclosure

We contacted all 14 CDN vendors to report all vulnerabilities found in our study. We provided them with detailed reports and mitigation solutions to eliminate the detected threats. Most vendors confirmed the vulnerabilities promptly and claimed to fix them as soon as possible. Some vendors have fixed the vulnerabilities, including Alibaba, G-core, and UPYun. The responses from CDN vendors are summarized below:

Alibaba evaluated the problem as a medium-risk vulnerability. They stated that they view it as indeed a problem for the CDN industry, and they would internally assess how to defend against it. They thanked us for our report and provided a reward of ≈$200.

G-core expressed their gratitude for our report and confirmed the vulnerability of their CDN. Subsequently, they have informed us that they have fixed the vulnerabilities.

UPYun thanked us for the report and discussed with us the details of the attack and its potential consequences. They have fixed the vulnerabilities.

Qiniu thanks for this research discovery and acknowledged the issues found in the paper. They would internally assess how to defend against it.

ChinaNetCenter expressed appreciation for our study and informed us that they are investigating to validate the attack scenarios and explore the effectiveness of mitigation already available.

Cachefly thanked us for our work and acknowledged the potential for abuse of these vulnerabilities but said that their denial of service protection mechanism blocks abusive traffic.

CloudFront acknowledged our report and stated that these behaviors align with the CDN's design to facilitate such actions.

Cloudflare thanked us for the report and is evaluating these vulnerabilities. However, they have provided no further response to date.

Azure and Baidu expressed gratitude for the report and acknowledged that the attacks resulted from misconfiguration.

Fastly acknowledged our report, but provided no further comment to date.

So far, we have disclosed the vulnerability reports to Bunny, CDNetworks, and Edgio, but we have not received any response. We are actively seeking further communication with them.

5.4 Anonymity and Cost

One may argue that launching these attacks in the real world is unlikely due to associated costs and the risk of exposing the attacker's identity. However, CDN vendors, presumably for competitive reasons, provide much convenience for their prospective customers (and thus for attackers). Table 11 shows the registration information required to begin using the free or free-trial services of the CDN vendors in our study. Seven out of fourteen CDN vendors require only a valid email address. Azure, Cachefly, and CloudFront require valid credit cards (could be gift cards or stolen). ChinaNetCenter, Qiniu, and UPYun require a valid phone number (could be anonymous). Alibaba requires users to verify their identity through a valid bank card, which takes an attacker more effort to keep anonymous.

Table 11: CDN registration requirements, cost, and origin verification.

	Register Requirements	Price	Origin Verification
Alibaba	C1, C2, C4	Free trial	No Verification
Azure	C1, C3	Free trial	No Verification
Baidu	C1	Free service	No Verification
Bunny	C1	Free trial	No Verification
Cachefly	C1, C3	Free service	No Verification
CDNetworks	C1	Free trial	No Verification
ChinaNetCenter	C1, C2	Free trial	No Verification
Cloudflare	C1	Free service	Domain Blacklist
CloudFront	C1, C3	Free trial	No Verification
Edgio	C1	Free trial	No Verification
Fastly	C1	Free service	No Verification
G-core	C1	Free service	No Verification
Qiniu	C1, C2	Free service	No Verification
UPYun	C1, C2	Free service	No Verification

† C1 means an Email address is required to register an account.
‡ C2 means a Phone number is required to register an account.
‖ C3 means a Credit card is required to register an account.
* C4 means a Bank card is required to register an account.

5.5 Severity Assessment

A severe and widespread practical influence. According to our experimental results, the `Image Optimization`, `Request Modification`, `Method Conversion`, and `Connection Decoupling` are nearly proportional to the size of the target resources, leading to the consumption of the origin server's outgoing bandwidth. Meanwhile, request modification attacks can exhaust the origin server's incoming bandwidth. As detailed in Section 3, the CDNs we tested enjoy global popularity and a high market share. These CDNs host 36.5% of the websites on the Tranco Top 10,000 list. Therefore, many well-known websites are vulnerable to our `BtOAmp` attack.

A low-cost and efficient DDoS attack. Unlike other DDoS attacks that need to control a large scale of botnets [37], the attacker only needs an ordinary laptop to launch the `BtOAmp` attacks. As the CDN nodes are dispersed globally and form a naturally distributed botnet [38, 39], it allows attackers to easily congest the target network and possibly cause a denial of service in seconds for a low cost.

A significant monetary loss to the victims. Most CDNs charge their website customers by traffic consumption, including Alibaba, Azure, Bunny, Baidu, Cloudflare, CloudFront, G-core [40–43]. Attackers can exploit back-to-origin strategies to launch a `BtOAmp` attack against CDN-powered websites, causing a very high CDN service fee and bandwidth fee to the website [44].

A security challenge to anti-DDoS. Traditional DDoS attacks consume bandwidth and primarily target the victim's incoming bandwidth. `BtOAmp` attacks can exhaust both the victim's incoming and outgoing bandwidth. Further, abusing the CDN to launch amplification attacks certainly will pose security challenges for detecting DDoS attacks [45, 46].

5.6 Limitations

There may be some limitations in this study. Our methodology for identifying vulnerabilities cannot guarantee the detection of all possible issues due to the complexity of the HTTP protocol and CDNs. We conducted tests on only 14 CDN vendors, but we believe that other CDN vendors may have similar vulnerabilities. Moreover, as CDNs continue to evolve, new vulnerabilities may emerge that were not previously detected.

6 Related Work

6.1 CDN Security

According to reports [47], nearly one-fifth of Internet traffic is transmitted through CDNs. Therefore, CDN infrastructure is the cornerstone of the Internet, and the security of CDNs has been extensively researched [48]. Due to the DDoS protection provided by CDNs [49, 50], the search for vulnerabilities in CDN architectures or implementations and attacks on CDN-hosted origin servers have been a hot topic in the field of network security.

Previous research has found various vulnerable implementations among CDN vendors. For example, Chen et al. [17] discovered inconsistency of HTTP request handling policies among CDN vendors, where attackers can chain CDN nodes into a loop, causing the malicious request to be processed repeatedly and reducing CDN's availability. Guo et al. [51] identified new architectural weaknesses in CDN infrastructure. Attackers could exploit CDNs to launch pulse attack waves on their hosted origin servers, disrupting their availability. Due to the conflicting nature of CDNs as intermediaries and the end-to-end encryption of HTTPS, previous researchers have explored TLS key management issues on CDN platforms, such as private key sharing and inefficient revocation [52, 53]. Furthermore, attackers can exploit disparities in the interpretation of HTTP header fields between CDNs and origin servers, thereby enabling the misuse of CDN caching mechanisms to initiate cache poisoning attacks [7], cache-poisoned-DoS attacks [54], and web cache deception attacks [55]. Furthermore, CDNs' high reputation and invisibility can be exploited to escape internet censorship, such as domain fronting [56], domain borrowing [57], and domain shadowing [36] that compete with the constantly-evolving censorship systems.

In summary, compared to previous CDN security work, considering the widespread use of CDNs for DDoS attack defense, we have identified implementation vulnerabilities in `back-to-origin` strategies of CDNs, proposed a new class of amplification attacks, and conducted real-world security assessments of fourteen popular CDNs. This research provides a valuable supplement to existing CDN security research.

6.2 Amplification Attacks

Amplification attacks constitute a well-established area of research within the realm of cybersecurity. The UDP protocol has been extensively researched for amplification attacks. Sieklik et al. [58] conducted a further analysis of amplification attacks based on DNSSEC, resulting in amplification factors of up to 44 times. Anagnostopoulos et al. [59] studied TFTP amplification attacks with amplification factors of 60 times. Booth et al. [60] revealed that UDP amplification attacks, recruiting UDP servers on the internet as reflectors, can achieve amplification factors up to 556 times. The TCP protocol can also be abused to launch an amplification attack. Kuhrer et al. [61] proposed that amplification attacks can be launched using the three-way handshake mechanism of the TCP protocol, and application-layer protocols based on the TCP protocol, e.g., FTP, Telnet, and SSH, are vulnerable to these attacks. Unfortunately, even the HTTP protocol has been abused. Li et al. [28] exposed that amplification attacks leveraging the HTTP Range Request mechanism can achieve amplification factors as high as 43000 times. Guo et al. [62] found that the compressed headers can be amplified when HTTP/2 downgrades to HTTP/1, which causes an amplification attack. Triukose et al. [31] proposed an attack that exhausts the origin server bandwidth by quickly disconnecting the client-CDN connection. We evaluated this attack and found that most CDNs mitigate it. However, our decoupled CDN connection attack can bypass this defense to attack the website powered by CDN.

In conclusion, our research reveals that leveraging `back-to-origin` strategies can launch a novel class of amplification attacks against websites hosted on CDNs. More significantly, when the website is hosted by CDN, although the CDN can defend against previously studied amplification attacks, our attacks can penetrate CDN DDoS protection and launch amplification attacks against the CDN-hidden websites.

7 Conclusion

The CDN has undeniably become an indispensable part of the internet landscape, providing several benefits, such as acting as a shield against DDoS attacks for websites hosted on CDN platforms. However, it is crucial to acknowledge that CDN infrastructure, particularly its `back-to-origin` strategies, can also be exploited, undermining the very DDoS protection it is supposed to provide.

We have presented a novel class of amplification attacks and their real-world evaluation of fourteen CDN vendors. This research reveals the flawed trade-off that CDN vendors made between performance and security. Experiment results show that attackers can exploit CDN `back-to-origin` strategies to successfully launch an amplification DoS attack against website servers. We envision our work being able to urge CDNs to raise their security standards and inspire more researchers to explore CDN-related security.

Acknowledgement

We sincerely thank all anonymous reviewers and our shepherd. Their insightful reviews and comments have improved this paper, with a special note of appreciation to our shepherd for their thoughtful and patient guidance. We also acknowledge the prompt response from CDN vendors, notably Alibaba, G-core, UPYun, and Qiniu, in fixing the vulnerabilities we found. This work is supported by the National Key Research and Development Program of China (Grant No. 2021YFB0301100), National Natural Science Foundation of China (No.62072109, No.62272265), and Natural Science Foundation of Fujian Province (No. 2021J06013).

References

[1] Cloudflare. what-is-a-cdn, 2023. https://developers.cloudflare.com/cache/concepts/cache-behavior/.

[2] Cloudflare. The cloudflare image optimization., 2023. https://developers.cloudflare.com/images/.

[3] Cloudflare. The cloudflare request header modification., 2023. https://developers.cloudflare.com/rules/transform/request-header-modification/.

[4] Cloudflare. The cloudflare url-rewrite., 2023. https://developers.cloudflare.com/rules/transform/url-rewrite/.

[5] Milad Ghaznavi, Elaheh Jalalpour, Mohammad A. Salahuddin, Raouf Boutaba, Daniel Migault, and Stere Preda. Content delivery network security: A survey. *IEEE Commun. Surv. Tutorials*, 23(4):2166–2190, 2021.

[6] Tencent. Tencent cloud security white paper, 2019. https://main.qcloudimg.com/raw/ea77661307adc3825990e159d851d406.pdf.

[7] Jianjun Chen, Jian Jiang, Hai-Xin Duan, Nicholas Weaver, Tao Wan, and Vern Paxson. Host of troubles: Multiple host ambiguities in HTTP implementations. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 1516–1527. ACM, 2016.

[8] Yossi Gilad, Amir Herzberg, Michael Sudkovitch, and Michael Goberman. Cdn-on-demand: An affordable ddos defense via untrusted clouds. In *NDSS*, 2016.

[9] Nishat I Mowla, Inshil Doh, and Kijoon Chae. Multi-defense mechanism against ddos in sdn based cdni. In *2014 Eighth International Conference on Innovative*

Mobile and Internet Services in Ubiquitous Computing, pages 447–451. IEEE, 2014.

[10] Jan Renz and Christoph Meinel. Improving mobile and worldwide performance through image optimization and distributed content delivery. In *EDULEARN18 Proceedings*, pages 5983–5991. IATED, 2018.

[11] Cloudflare. The cloudflare handling head requests., 2023. https://developers.cloudflare.com/cache/concepts/cache-behavior/.

[12] Datanyze. Content delivery networks market share report, 2023. https://www.datanyze.com/market-share/cdn--10/.

[13] Lin Jin, Shuai Hao, Haining Wang, and Chase Cotton. Your remnant tells secret: Residual resolution in ddos protection services. In *48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2018, Luxembourg City, Luxembourg, June 25-28, 2018*, pages 362–373. IEEE Computer Society, 2018.

[14] Lin Jin, Shuai Hao, Haining Wang, and Chase Cotton. Unveil the hidden presence: Characterizing the backend interface of content delivery networks. In *27th IEEE International Conference on Network Protocols, ICNP 2019, Chicago, IL, USA, October 8-10, 2019*, pages 1–11. IEEE, 2019.

[15] Shuai Hao, Yubao Zhang, Haining Wang, and Angelos Stavrou. End-users get maneuvered: Empirical analysis of redirection hijacking in content delivery networks. In William Enck and Adrienne Porter Felt, editors, *27th USENIX Security Symposium, USENIX Security 2018, Baltimore, MD, USA, August 15-17, 2018*, pages 1129–1145. USENIX Association, 2018.

[16] Tranco List, 2024. https://tranco-list.eu/.

[17] Jianjun Chen, Xiaofeng Zheng, Hai-Xin Duan, Jinjin Liang, Jian Jiang, Kang Li, Tao Wan, and Vern Paxson. Forwarding-loop attacks in content delivery networks. In *23rd Annual Network and Distributed System Security Symposium, NDSS 2016, San Diego, California, USA, February 21-24, 2016*. The Internet Society, 2016.

[18] Run Guo, Jianjun Chen, Baojun Liu, Jia Zhang, Chao Zhang, Hai-Xin Duan, Tao Wan, Jian Jiang, Shuang Hao, and Yaoqi Jia. Abusing cdns for fun and profit: Security issues in cdns' origin validation. In *37th IEEE Symposium on Reliable Distributed Systems, SRDS 2018, Salvador, Brazil, October 2-5, 2018*, pages 1–10. IEEE Computer Society, 2018.

[19] Cloudflare. What is an image optimizer? | how to reduce image sizes, 2023. https://www.cloudflare.com/learning/performance/glossary/what-is-an-image-optimizer/.

[20] Sachin Dhawan. A review of image compression and comparison of its algorithms. *International Journal of electronics & Communication technology*, 2(1):22–26, 2011.

[21] Hui Zeng, Lida Li, Zisheng Cao, and Lei Zhang. Reliable and efficient image cropping: A grid anchor based approach. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5949–5957, 2019.

[22] tcpdump, 2023. https://www.tcpdump.org/.

[23] Sharon Lim, J Ha, H Kim, Y Kim, and S Yang. A sdn-oriented ddos blocking scheme for botnet-based attacks. In *2014 Sixth International Conference on Ubiquitous and Future Networks (ICUFN)*, pages 63–68. IEEE, 2014.

[24] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. Understanding the mirai botnet. In *26th USENIX security symposium (USENIX Security 17)*, pages 1093–1110, 2017.

[25] Esraa Alomari, Selvakumar Manickam, Brij Bhooshan Gupta, Shankar Karuppayah, and Rafeef Alfaris. Botnet-based distributed denial of service (ddos) attacks on web servers: classification and art. *arXiv preprint arXiv:1208.0403*, 2012.

[26] Cloudflare. Http request header modification rules, 2023. https://developers.cloudflare.com/rules/transform/request-header-modification/.

[27] Balachander Krishnamurthy, Craig Wills, and Yin Zhang. On the use and performance of content distribution networks. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, pages 169–182, 2001.

[28] Weizhong Li, Kaiwen Shen, Run Guo, Baojun Liu, Jia Zhang, and Haixin Duan et al. CDN backfired: Amplification attacks based on HTTP range requests. In *50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2020, Valencia, Spain, June 29 - July 2, 2020*, pages 14–25. IEEE, 2020.

[29] Tanvir Habib Sardar and Zahid Ansari. Detection and confirmation of web robot requests for cleaning the voluminous web log data. In *2014 International Conference on the IMpact of E-Technology on US (IMPETUS)*, pages 13–19. IEEE, 2014.

[30] Craig Pratt, Darshak Thakore, and Barbara Stark. HTTP Random Access and Live Content. RFC 8673, November 2019. https://www.rfc-editor.org/info/rfc8673.

[31] Sipat Triukose, Zakaria Al-Qudah, and Michael Rabinovich. Content delivery networks: Protection or threat? In *Computer Security - ESORICS 2009, 14th European Symposium on Research in Computer Security, Saint-Malo, France, September 21-23, 2009. Proceedings*, volume 5789 of *Lecture Notes in Computer Science*, pages 371–389. Springer, 2009.

[32] Roy T. Fielding, Mark Nottingham, and Julian Reschke. HTTP/1.1. RFC 9112, June 2022. https://www.rfc-editor.org/info/rfc9112.

[33] Bahruz Jabiyev, Steven Sprecher, Kaan Onarlioglu, and Engin Kirda. T-reqs: Http request smuggling with differential fuzzing. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1805–1820, 2021.

[34] Jack Jansen, Shishir Subramanyam, Romain Bouqueau, Gianluca Cernigliaro, Marc Martos Cabré, Fernando Pérez, and Pablo Cesar. A pipeline for multiparty volumetric video conferencing: transmission of point clouds over low latency dash. In *Proceedings of the 11th ACM Multimedia Systems Conference*, pages 341–344, 2020.

[35] Behnam Shobiri, Mohammad Mannan, and Amr Youssef. Cdns' dark side: Security problems in cdn-to-origin connections. *Digital Threats*, 4(1), mar 2023.

[36] Mingkui Wei. Domain shadowing: Leveraging content delivery networks for robust blocking-resistant communications. In *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 3327–3343. USENIX Association, 2021.

[37] Esraa Alomari, Selvakumar Manickam, Brij B. Gupta, Shankar Karuppayah, and Rafeef Alfaris. Botnet-based distributed denial of service (ddos) attacks on web servers: Classification and art. *CoRR*, abs/1208.0403, 2012.

[38] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. Understanding the mirai botnet. In *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017*, pages 1093–1110. USENIX Association, 2017.

[39] Anestis Karasaridis, Brian Rexroad, and David A. Hoeflin. Wide-scale botnet detection and characterization. In *First Workshop on Hot Topics in Understanding Botnets, HotBots'07, Cambridge, MA, USA, April 10, 2007*. USENIX Association, 2007.

[40] Alibaba Cloud. Content delivery network pricing & purchasing methods, 2023. https://www.alibabacloud.com/zh/product/content-deliver-network/pricing?spm=a3c0i.7938564.220486.75.26d62aecnaGFXf.

[41] Azure. Content delivery network pricing, 2023. https://azure.microsoft.com/en-us/pricing/details/cdn/.

[42] Amazon CloudFront. Amazon cloudfront pricing, 2023. https://aws.amazon.com/tw/cloudfront/pricing/.

[43] G-core. edge-network pricing, 2023. https://gcore.com/pricing/edge-network.

[44] Cloudflare. Cdn performance bundle pricing, 2023. https://www.cloudflare.com/lp/pg-cdn-performance-bundle.

[45] Zaoxing Liu, Hun Namkung, Georgios Nikolaidis, Jeongkeun Lee, Changhoon Kim, Xin Jin, Vladimir Braverman, Minlan Yu, and Vyas Sekar. Jaqen: A high-performance switch-native approach for detecting and mitigating volumetric ddos attacks with programmable switches. In Michael Bailey and Rachel Greenstadt, editors, *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*, pages 3829–3846. USENIX Association, 2021.

[46] Zhijun Wu, Wenjing Li, Liang Liu, and Meng Yue. Low-rate dos attacks, detection, defense, and challenges: A survey. *IEEE Access*, 8:43920–43943, 2020.

[47] Data Economy. Data economy frontline. edge computing as the most misunderstood weapon of the iot world, 2019. https://www.clearblade.com/press/data-economyfrontline-edge-computing-as-the-most-misunderstood-weapon-of-the-iot-world/.

[48] Michele Luglio, Simon Pietro Romano, Cesare Roseti, and Francesco Zampognaro. Service delivery models for converged satellite-terrestrial 5g network deployment: A satellite-assisted CDN use-case. *IEEE Netw.*, 33(1):142–150, 2018.

[49] Mohammed Imthiyas, Sharyar Wani, Rawad AbdulKhaleq Abdulmolla Abdulghafor, Adamu Abubakar Ibrahim, and Abudl Hafeez Mohammad. Ddos mitigation: A review of content delivery network and its ddos

defence techniques. *International Journal on Perceptive and Cognitive Computing*, 6(2):67–76, 2020.

[50] Elaheh Jalalpour, Milad Ghaznavi, Daniel Migault, Stere Preda, Makan Pourzandi, and Raouf Boutaba. A security orchestration system for cdn edge servers. In *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pages 46–54. IEEE, 2018.

[51] Run Guo, Jianjun Chen, Yihang Wang, Keran Mu, Baojun Liu, Xiang Li, Chao Zhang, Haixin Duan, and Jianping Wu. Temporal cdn-convex lens: A cdn-assisted practical pulsing ddos attack. In *32nd USENIX Security Symposium, USENIX Security 2023, Anaheim, CA, USA, August 9-11, 2023*. USENIX Association, 2023.

[52] Jinjin Liang, Jian Jiang, Hai-Xin Duan, Kang Li, Tao Wan, and Jianping Wu. When HTTPS meets CDN: A case of authentication in delegated service. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 67–82. IEEE Computer Society, 2014.

[53] Frank Cangialosi, Taejoong Chung, David R. Choffnes, Dave Levin, Bruce M. Maggs, Alan Mislove, and Christo Wilson. Measurement and analysis of private key sharing in the HTTPS ecosystem. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 628–640. ACM, 2016.

[54] Hoai Viet Nguyen, Luigi Lo Iacono, and Hannes Federrath. Your cache has fallen: Cache-poisoned denial-of-service attack. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 1915–1936. ACM, 2019.

[55] Seyed Ali Mirheidari, Sajjad Arshad, Kaan Onarlioglu, Bruno Crispo, Engin Kirda, and William Robertson. Cached and confused: Web cache deception in the wild. In *29th USENIX Security Symposium, USENIX Security 2020, August 12-14, 2020*, pages 665–682. USENIX Association, 2020.

[56] David Fifield, Chang Lan, Rod Hynes, Percy Wegmann, and Vern Paxson. Blocking-resistant communication through domain fronting. *Proc. Priv. Enhancing Technol.*, 2015(2):46–64, 2015.

[57] Junyu Zhou and Tianze Ding. Domain borrowing: Catch my c2 traffic if you can, 2021. https://i.blackhat.com/asia-21/Thursday-Handouts/as-21-Ding-Domain-Borrowing-Catch-My-C2-Traffic-If-You-Can.pdf.

[58] Marios Anagnostopoulos, Georgios Kambourakis, Panagiotis Kopanos, Georgios Louloudakis, and Stefanos Gritzalis. DNS amplification attack revisited. *Comput. Secur.*, 39:475–485, 2013.

[59] Boris Sieklik, Richard Macfarlane, and William J. Buchanan. Evaluation of TFTP ddos amplification attack. *Comput. Secur.*, 57:67–92, 2016.

[60] Todd G. Booth and Karl Andersson. Elimination of dos UDP reflection amplification bandwidth attacks, protecting TCP services. In *Future Network Systems and Security - First International Conference, FNSS 2015, Paris, France, June 11-13, 2015, Proceedings*, volume 523 of *Communications in Computer and Information Science*, pages 1–15. Springer, 2015.

[61] Marc Kührer, Thomas Hupperich, Christian Rossow, and Thorsten Holz. Hell of a handshake: Abusing TCP for reflective amplification ddos attacks. In *8th USENIX Workshop on Offensive Technologies, WOOT '14, San Diego, CA, USA, August 19, 2014*. USENIX Association, 2014.

[62] Run Guo, Weizhong Li, Baojun Liu, Shuang Hao, Jia Zhang, Haixin Duan, Kaiwen Sheng, Jianjun Chen, and Ying Liu. CDN judo: Breaking the CDN dos protection with itself. In *27th Annual Network and Distributed System Security Symposium, NDSS 2020, San Diego, California, USA, February 23-26, 2020*. The Internet Society, 2020.

Table 12: Techniques to launch Image Optimization attack

	image compression	image cropping	combination
Alibaba	?image_process=format,webp	?image_process=crop,center,w_1,h_1	?image_process=crop,center,w_1,h_1/format,webp
Bunny	?quality=1	?crop=1,1	
Cachefly	?format(webp)	?width:1xheight:1	
CDNetworks	?f=webp	?crop=p_1,w_1,h_1	?f=webp&crop=p_1,w_1,h_1
ChinaNetCenter	?&output=webp	?&w=1&h=1	?&w=1&h=1&output=webp
Cloudflare	/cdn-cgi/image/format=webp	/cdn-cgi/image/width=1	
CloudFront	?format=webp	?width=1&hight=1	?format=webp&width=1&hight=1
Edgio	?auto=webp,smallest	?width=1&height=1	?width=1&height=1&fit=crop&format=webp
Fastly	?format=webp	?crop=1,1	?crop=1.1&format=webp
G-core	?fmt=webp	?crop=1,1	
Qiniu	?imageMogr2/format/webp	?imageMogr2/crop/1x1	?imageMogr2/crop/1x1/format/webp
UPYun	!/format/webp	!/fwfh/1x1	!/fwfh/1x1/format/webp

A Appendix

A.1 Details of Image Optimization Attack

To elaborate on how to launch an `Image Optimization` attack, we summarize the parameters for launching an image optimization attack in 12. Taking Fastly as an example, for example, utilizing the parameter "?format=webp" can trigger the CDN to compress the image into webp format. For example, the parameter "?crop=1,1" can trigger the CDN to crop the requested image to one pixel. Moreover, attackers can use the parameter "?crop=1.1&format=webp" to exploit the CDN to compress the image into webp format and crop it to one pixel.