CDN Backfired: Amplification Attacks Based on HTTP Range Requests

Weizhong Li*, Kaiwen Shen*, Run Guo*, Baojun Liu*, Jia Zhang*¶✉,
Haixin Duan*‖✉, Shuang Hao†, Xiarun Chen‡, Yao Wang§
* Tsinghua University,
{lwz17, skw17, gr15, lbj15}@mails.tsinghua.edu.cn, zhangjia@cernet.edu.cn, duanhx@tsinghua.edu.cn
† University of Texas at Dallas, shao@utdallas.edu
‡ Peking University, xiar_c@pku.edu.cn
§ Beijing Information Science & Technology University, lemonvegetablefish@gmail.com
¶ Beijing National Research Center for Information Science and Technology
‖ Research Institute of Qi-AnXin Group

***Abstract*—Content Delivery Networks (CDNs) aim to improve network performance and protect against web attack traffic for their hosting websites. And the HTTP range request mechanism is majorly designed to reduce unnecessary network transmission. However, we find the specifications failed to consider the security risks introduced when CDNs meet range requests.**

In this study, we present a novel class of HTTP amplification attack, Range-based Amplification (RangeAmp) Attacks. It allows attackers to massively exhaust not only the outgoing bandwidth of the origin servers deployed behind CDNs but also the bandwidth of CDN surrogate nodes. We examined the RangeAmp attacks on 13 popular CDNs to evaluate the feasibility and real-world impacts. Our experiment results show that all these CDNs are affected by the RangeAmp attacks. We also disclosed all security issues to affected CDN vendors and already received positive feedback from all vendors.

***Index Terms*—CDN Security, HTTP Range Request, Amplification Attack, DDoS**

I. Introduction

Content Delivery Networks (CDNs) redirect web requests from client users to geographically distributed surrogate servers and are regarded as an important part of the Internet infrastructure. CDN vendors significantly improve the performance and scalability of their hosting websites by delivering their web resources globally. In addition, CDNs are also famous for their sophisticated protection mechanisms against web attacks, including normalizing or filtering the intrusions traffic, offloading DDoS traffic to global surrogate nodes. As a result, CDN vendors are widely trusted by the most popular websites all over the world. For example, Akamai, the leading CDN service provider, is responsible for serving between 15% and 30% of all web traffic according to a public report [1].

At the same time, the HTTP protocol also goes further. The HTTP range request mechanism is designed to allow a client to request just a part of a web resource [2]. Therefore, the client can not only retrieve partial content of large representations but also efficiently recover from partially failed transfers. Currently, despite the fact that this mechanism is only an optional feature of HTTP, the RFC specifications still suggest that web servers and intermediate cache servers should support it. And in the real world, range requests have been strongly supported by CDN vendors and widely applied in multi-thread file downloading and resuming from break-point.

Unfortunately, while the RFC specifications are generally clear on how to parse and interpret range requests, we find the implementations of CDN vendors problematic. In this study, we present two types of "Range-based Amplification (RangeAmp) Attacks", which allow attackers to exploit the `Range` implementation vulnerabilities and damage DDoS protection mechanisms of CDNs. Specifically, the RangeAmp attacks include Small Byte Range (SBR) Attack and Overlapping Byte Range (OBR) Attack. The SBR attack, which leverages the aggressive prefetch strategy of CDN platforms, enables attackers to massively consume network bandwidth of the origin servers hosted on CDNs by performing some crafted HTTP range requests, see section IV-B. Worse, by exploiting the implementation flaws on multi-range requests and by connecting the vulnerable CDNs, the OBR attack even allows attackers to directly damage the performance of CDN nodes by building up a huge number of multi-part responses between specific CDN nodes, see section IV-C. Therefore, the RangeAmp attacks can bring significantly detrimental impacts on both CDN hosting servers and CDN surrogate nodes.

In this study, we also evaluate the RangeAmp attacks in the wild by conducting a series of controlled experiments on 13 popular CDN vendors. Our experiment results show that all examined CDN vendors are seriously affected by the RangeAmp attacks, with 13 CDNs vulnerable to the SBR attack and 11 combinations of cascaded CDNs vulnerable to the OBR attack. For instance, using Akamai or G-Core Labs to perform an SBR attack, an attacker is able to compel the origin website to generate response traffic *43000 times* larger than the one received by the attacker. Besides, when connecting Cloudflare and Akamai to launch an OBR attack and selecting a 1KB file as the target resource, an attacker is able to force specific nodes of these two CDNs to transfer traffic over 12MB with just one multi-range request. Therefore, the RangeAmp attacks introduce serious security threats against

✉ Corresponding author.

978-1-7281-5809-9/20/$31.00 ©2020 IEEE
DOI 10.1109/DSN48063.2020.00022

the availability of CDN infrastructure.

At last, we propose mitigation solutions and recommendations to different roles, including the origin website administrators, the CDN vendors, and the HTTP protocol specifications. We also responsibly disclosed all found vulnerabilities to affected CDN vendors. Until the paper was finalized, we received some positive feedback from all vendors, some of which have fixed RangeAmp vulnerabilities.

Overall, our study makes the following contributions.

- We present a novel class of HTTP amplification attack, Range-based Amplification (RangeAmp) Attacks. The RangeAmp attacks can be used to consume the outgoing bandwidth of victims, which not only downgrades the network availability but also brings economic losses.
- We examine the RangeAmp attacks on 13 popular CDN vendors and evaluate the feasibility and severity of RangeAmp vulnerabilities. We find all examined CDNs are vulnerable to the RangeAmp attacks, and the amplification factor is up to 43000 times in some cases.
- We also responsibly disclosed all security issues to affected CDN vendors. Further, we analyze the root cause of RangeAmp vulnerabilities and propose countermeasures and mitigation solutions.

We organize the rest of this paper as follows. Section II provides the background of CDN and HTTP range request mechanism. Section III shows the range-specific implementations in CDNs. Section IV describes the details of the RangeAmp attacks. We evaluate the feasibility of the RangeAmp attacks and explore the amplification factors in Section V. We discuss mitigation solutions and our responsible disclosure in Section VI and Section VII. Section VIII elaborates on the related works, including HTTP Range security, CDN security, and amplification attacks. And section IX concludes this paper.

II. Background

A. CDN Overview

CDN network is made up of server clusters geo-located globally. CDNs have evolved to become an important part of the Internet infrastructure. It not only improves the performance for its customer websites but also provides security features such as DDoS protection mechanisms.

CDN network can be divided into two parts: central and edge nodes. Central nodes are used for global load balancing and content management. Edge nodes are used for content distribution and caching and classified as ingress nodes and egress nodes according to their positions and functions. In general, ingress nodes are close to the user and responsible for user access and content distribution, while egress nodes are close to the origin website and retrieve contents from it.

As it is shown in Fig 1, in a CDN environment, there are multiple segments of connectivity in the network path between the client and the origin server, including one between the client and the CDN (recognized as *client-cdn*), one between the CDN and the origin server (recognized as *cdn-origin*), and one(s) within the CDN or between CDNs.

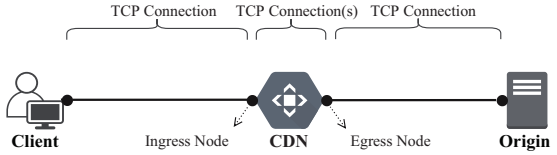


Fig. 1. Multiple segments of connectivity in a CDN environment

Two CDNs can be cascaded together [3], as shown in Fig 3b. For convenience, we recognize the front-end CDN as FCDN and the back-end CDN as BCDN. There are at least 3 TCP connections in Figure 3b, including *client-fcdn*, *fcdn-bcdn* and *bcdn-origin*.

If a user requests data, the CDN first tries to respond from the local cache [4]. In the case of a cache miss, it forwards the request to the origin server to obtain the target resource and caches the response for subsequent requests. This mechanism can efficiently reduce user access delay and decrease load pressure on the origin server. Besides, CDNs select edge nodes dynamically by load balancing, which provides DDoS protection to the origin server.

However, **a user is able to make a cache miss and force web requests to be forwarded to the origin server**. Previous studies [5], [6] show that dynamic resource requests will be always forwarded to the origin server, and appending a random query string into the target URL can also bypass the CDN's caching mechanism. Moreover, most CDNs (*e.g.*, Azure, Cloudflare, etc) provide configurable options to customize caching policy [3], which makes a malicious customer able to disable resource caching.

B. HTTP Range Request Mechanism

Because of canceled requests or dropped TCP connections, HTTP clients often encounter interrupted data transfers. However, HTTP is a stateless application protocol. When requesting large media or downloading files, interrupted transfers require the client to re-transfer the entire resource. This brings inefficient network transmissions and bad user experiences.

Therefore, the protocol specifications [2], [7], [8] introduce the HTTP range request mechanism to improve the transmission efficiency of web resources. Range requests allow clients to efficiently recover from partially failed transfers and retrieve partial content of large resources, effectively reducing unnecessary data transmission. This mechanism is especially useful to perform multi-thread transfers and resuming from break-point when downloading large files.

Although the range request mechanism is an optional feature of HTTP, the specifications suggest that origin servers and intermediate caches ought to support it when possible. If a server supports byte-range requests, it will insert an `Accept-Ranges` header in the response and set the field value to "bytes", otherwise, it will set the field value to "none" or not insert such a response header.

A range request uses a `Range` header to specify one or more sub-ranges of the target resource, as shown in Figure 2a and Figure 2b. According to the specifications, the valid

```
GET /1KB.jpg HTTP/1.1
Host: example.com
Range: bytes=0-0
```

(a) range request with a single byte range

```
GET /1KB.jpg HTTP/1.1
Host: example.com
Range: bytes=1-1,-2
```

(b) range request with multiple byte ranges

```
HTTP/1.1 206 OK
Content-Length: 1
Accept-Ranges: bytes
Content-Type: image/jpeg
Content-Range: bytes 0-0/1000

\xff
```

(c) 206 response to the request in (a)

```
HTTP/1.1 206 OK
Content-Length: 208
Accept-Ranges: bytes
Content-Type: multipart/byteranges;
boundary=THIS_STRING_SEPARATES


--THIS_STRING_SEPARATES
Content-Type: image/jpeg
Content-Range: bytes 1-1/1000

\xff
--THIS_STRING_SEPARATES
Content-Type: image/jpeg
Content-Range: bytes 998-999/1000

f\x00
--THIS_STRING_SEPARATES--
```

(d) multipart response to the request in (b)

Fig. 2. Examples of range requests and partial responses

format of a `Range` header is "Range: bytes=first_byte_pos-[last_byte_pos]" or "Range: bytes=-suffix_length".

A website server can behave differently when receiving a range request: 1) If the server does not support range requests, it ignores the `Range` header and returns an HTTP 200 response when the request has no errors. Otherwise, 2) if the specified range is valid, it returns an HTTP 206 response; 3) if the `Range` header is invalid or the specified range is out of bounds, it returns an HTTP 416 response.

Based on the specified valid range, the server generates a single-part or multi-part 206 response. A single-part 206 response contains a `Content-Range` header to indicate where the transmitted partial content is located in the target resource, as shown in Figure 2c. A multi-part 206 response must contain a `Content-Type` header whose field value is "multipart/byteranges", indicating that it will be sent as a multi-part message. But it must not directly contain a `Content-Range` header, which will be sent in each part instead, as shown in Figure 2d.

III. Range-specific Implementations in CDNs

In this section, we first present why we specifically explore these 13 CDN providers. Then, we analyze and clarify their range request handling behaviors which lead to the RangeAmp attacks.

A. Consideration in Selecting CDN Vendors

We test 13 popular CDNs around the world, including Akamai, Alibaba Cloud, Azure, CDN77, CDNsun, Cloudflare, CloudFront, Fastly, G-Core Labs, Huawei Cloud, KeyCDN, StackPath, and Tencent Cloud. These CDNs are often studied in previous related works [3], [9], [10], and most of them rank high in the market share [11]. Moreover, most of these CDNs provide free or free-trial accounts, which indicates little cost to launch an attack.

Akamai only provides services for enterprise customers, but we manage to configure an Akamai service on the Microsoft Azure platform and have a free trial for one month. We check all ingress and egress IPs in corresponding Akamai experiments and confirmed that these IPs indeed belong to Akamai. Tencent Cloud only provides paid services, but it gives away 50GB of free traffic every month within half a year. Neither Huawei Cloud nor Alibaba Cloud provides free services, and we have spent less than $10 in our experiments.

In all subsequent experiments, we deploy our origin server individually behind these CDNs and apply their default configuration.

B. Differences in CDNs Handling Range Requests

According to HTTP specifications [2], [7], [8], HTTP implementations ought to support range requests when possible. To find out which CDNs support range requests, we invalidate it on our origin server and send a valid range request to each CDN. The result is that our origin server always returns a 200 response with no `Accept-Range` header, but all CDNs return a 206 response with an `Accept-Range` header whose field value is "bytes". Therefore, we conclude that these 13 CDNs all support range requests, indeed following the suggestion of the specifications.

However, it is not clearly defined in the specifications how CDNs should forward a range request. We find that CDNs have different policies to handle the `Range` header before forwarding a valid range request, including:

- *Laziness* – Forward the `Range` header without change.
- *Deletion* – Remove the `Range` header directly.
- *Expansion* – Extend it to a larger scale of byte range.

When receiving a range request, **most CDNs prefer to adopt the *Deletion* policy or the *Expansion* policy** (see Section V-A) because they believe that the client may continue requesting other byte ranges of the same resource. In this case, the CDN removes the `Range` header or extends it to a larger byte range when forwarding a range request, and then caches the responses for subsequent range requests. This does optimize caching, reduce access latency, and prevent excessive back-to-origin requests.

The range request mechanism also allows the client to request multiple sub-ranges of the target resource, as described in Section II-B. However, RFC2616 [7] places no restrictions on such multi-range requests. The "Apache Killer" [12], known as CVE-2011-3192 [13], can exhaust memory on the Apache server by creating a number of threads that use a `Range` header with multiple ranges. Therefore, RFC7233 [2] adds some security considerations to multi-range requests, suggesting that **an HTTP server ought to ignore, coalesce, or reject range requests with more than two overlapping ranges** or many small ranges in the `Range` header. We find that most CDNs indeed adopt the suggestion of RFC7233 **but unfortunately, some CDNs ignore it** (see Section V-A).

IV. Range-based HTTP Amplification Attacks

The *Deletion* and *Expansion* policy are beneficial for CDNs to improve service performance. But we notice that these policies will require CDNs to retrieve many more bytes from the origin server than the ones requested by the client. Also, if a CDN returns a multi-part response to a multi-range request without checking if ranges overlap, the response sent by the

CDN can be thousands of times larger than the one from the origin server. These cases will cause serious traffic differences between different connections in the network path from the client to the origin server.

A. Threat Model

The significant traffic differences caused by range-specific policies will bring a novel class of traffic amplification attacks, denoted "Range-based Amplification (RangeAmp) Attacks". We identified two scenarios of the RangeAmp attacks and respectively present them in Section IV-B and Section IV-C.

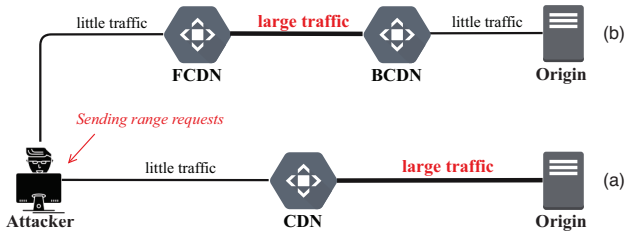


Fig. 3. *General construction of the RangeAmp Attacks*

In a RangeAmp attack, the attacker is able to craft malicious but legal range requests to the CDN, as shown in Fig 3. One of the victims is the origin server in Fig 3a, which is being normally hosted on the CDN by the owner, or maliciously deployed on the CDN by the attacker [14]. The other victims are the FCDN and the BCDN in Fig 3b, which is maliciously cascaded together by the attacker.

Through an empirical study, we show that the attacker can perform a traffic amplification attack with little cost and exhaust the bandwidth of its victims.

B. Small Byte Range(SBR) Attack

If a CDN adopts the *Deletion* or *Expansion* policy to handle range requests, an attacker can craft a `Range` header with a small byte range to launch a RangeAmp attack. We call it "Small Byte Range(SBR) Attack". In an SBR attack, the *cdn-origin* connection will transport a much larger traffic than the *client-cdn* connection, which makes the attacker able to attack against the origin server hosted on the CDN.

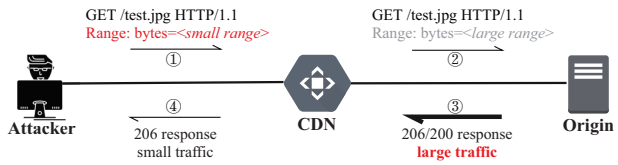


Fig. 4. *Flow and example construction of the SBR Attack*

As shown in Fig 4, the attacker crafts a range request with a small byte range like "Range: bytes=0-0", and sends it to a vulnerable CDN. As described at the end of Section II-A, an attacker can easily make a cache miss. Therefore, the CDN will remove the `Range` header or extend it to a larger byte range, and then forward the request to the origin server. This results in that the origin server returns an entire copy or a large range of the target resource, but the CDN returns a partial content with only the specified range, which can even be a single byte.

In an SBR attack, response traffic in the *client-cdn* connection is just hundreds of bytes (little). If the CDN adopts the *Deletion* policy, response traffic in the *cdn-origin* connection is equivalent to the entire target resource (much greater). Therefore, **the bigger the target resource, the larger the amplification factor.** But if the CDN adopts the *Expansion* policy, the amplification factor will only be a fraction of the one in the previous case.

C. Overlapping Byte Ranges(OBR) Attack

If the FCDN adopts the *Laziness* policy and the BCDN returns a multi-part response without checking whether ranges overlap, an attacker can craft a `Range` header with multiple overlapping byte ranges to launch another RangeAmp attack. We call it "Overlapping Byte Ranges(OBR) Attack". In an OBR attack, the *fcdn-bcdn* connection will transport a much larger traffic than the *bcdn-origin* connection, which makes the attacker able to greatly consume the bandwidth available between the FCDN and the BCDN. The attacker can send all multi-range requests to the same ingress node of the FCDN, and set the FCDN's origin server to be a specific ingress node of the BCDN, to perform the OBR attack against these specific nodes.

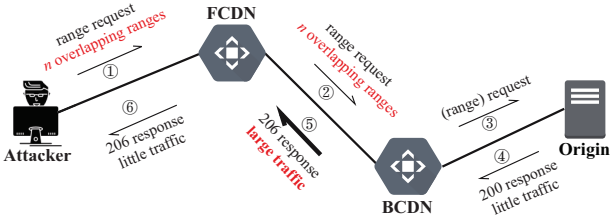


Fig. 5. *Flow and example construction of the OBR Attack*

As shown in Fig 5, the attacker crafts a multi-range request with n overlapping byte ranges like "Range: bytes=0-,0-,...,0-" (the number of "0-" is n), and sends it to the FCDN. The FCDN directly forwards it to the BCDN. After handling the `Range` header, the BCDN forwards the request to the origin server where range requests are disabled by the attacker. The origin server will return a 200 response with the entire copy of the target resource, but the BCDN will return a n-part response, which can be up to n times the size of the entire target resource.

The OBR attacker can set a small *TCP Receive Window* to make himself only receive little data [15], [16]. Besides, some CDNs will maintain the connection between itself and the upstream server when the *client-cdn* connection is abnormally aborted [5], such as CDNsun and CDN77. Thus, the attacker is able to consume much smaller resources by actively aborting the *client-cdn* connection.

In an OBR attack, when the target resource is fixed, response traffic in the *bcdn-origin* connection is always roughly

the same. But response traffic in the *fcdn-bcdn* connection is nearly proportional to the number of overlapping ranges. Apparently, **the greater the number of overlapping ranges, the larger the amplification factor.** But the number of overlapping ranges is limited by the maximum length of the `Range` header which is generally restricted by the request header size limit of particular CDN. Therefore, the maximum length of the `Range` header finally determines the upper-bound of the amplification factor.

V. Real-world Evaluation

To explore the feasibility and severity of RangeAmp vulnerabilities in the wild, we conduct a series of experiments. We examine which CDNs are vulnerable to the RangeAmp attacks, calculate the actual amplification factors, and analyze the practical impacts. In all experiments, our origin server is the same Linux server with 2.4GHz of CPU, 16G of memory and 1000Mbps of bandwidth. And our origin website is powered by Apache/2.4.18 with the default configuration applied.

A. Feasibility of the RangeAmp Attacks

To analyze whether RangeAmp vulnerabilities exist in practical environments, we test the actual range-specific policies of each CDN to figure out which CDNs are vulnerable to the SBR and/or OBR attack.

In our first experiment, the data-set is a large number of valid range requests automatically generated based on the ABNF rules described in the RFCs [2], [7], [8]. We send these range requests to each CDN and ensure that they will be forwarded to our origin server. At the same time, we collect all requests and responses on the client and the origin server.

We compare the request sent by the client with the corresponding one received by the origin server to analyze the range forwarding behaviors of each CDN, and the vulnerable results of small byte range(s) and multiple overlapping ranges are summarized respectively in Table I and Table II. We also compare the payload size of the response sent by the server and the corresponding one received by the client to discover the vulnerable replying behaviors of multi-range requests, as shown in Table III.

Table I shows that a total of 13 CDNs are vulnerable to the SBR attack. The second column lists the vulnerable range formats, and the third column presents the CDNs' actual policies of handling the corresponding `Range` headers. The details are shown below:

1) **Akamai**, **Alibaba Cloud**, **CDN77**, **CDNsun**, **Cloudflare**, **Fastly**, **G-Core Labs**, **Huawei Cloud**, and **Tencent Cloud** adopt the *Deletion* policy for some formats of the `Range` header. Among them, the entries with (*) are conditional. Alibaba Cloud and Tencent Cloud both provide a *Range* option to configure whether the back-to-origin request contains a `Range` header, and only when this option is set to *disable*, the vulnerable range forwarding behaviors shown in Table I occurs. Huawei Cloud also provides such a *Range* option, but it is vulnerable only when this option is set to *enable*. Cloudflare allows its users to customize caching rules, and it is vulnerable only when the target path is configured to be cacheable.

2) **Azure** first adopts the *Deletion* policy to handle a `Range` header like "Range: bytes=first-last". But if the file size of the target resource is larger than 8MB and [first,last]$\subset$[8388608,16777215], Azure will adopt the *Expansion* policy to replace the `Range` header with "Range: bytes=8388608-16777215" and then forward the new request to the origin server. In this case, there will be two *cdn-origin* connections, and if the HTTP payload transferred in the first *cdn-origin* connection is over 8MB, Azure will close this connection immediately. Considering network latency, actual response traffic in the first connection will be a little larger than 8MB. As a result, if the target resource exceeds 16MB, the response traffic in the two *cdn-origin* connections will be both approximately 8MB.

TABLE I
Range Forwarding Behaviors vulnerable to SBR Attack

CDN	Vulnerable Range Format	Forwarded Range Format
Akamai	bytes=first-last bytes=-suffix	None None
Alibaba Cloud	bytes=-suffix	None (*)
Azure	bytes=first-last (F$\leq$8MB) bytes=8388608-8388608 (F$>$8MB)	None None & bytes=8388608-16777215
CDN77	bytes=first-last (first$<$1024)	None
CDNsun	bytes=0-last	None
Cloudflare	bytes=first-last bytes=-suffix	None (*) None (*)
CloudFront	bytes=first-last bytes=$first_1$-$last_1$,...,$first_n$-$last_n$	bytes=$first'$-$last'$ bytes=$first'$-$last'$
Fastly	bytes=first-last bytes=-suffix	None None
G-Core Labs	bytes=first-last bytes=-suffix	None None
Huawei Cloud	bytes=-suffix (F$<$10MB) bytes=first-last (F$\geq$10MB)	None (*) None & None (*)
KeyCDN	bytes=first-last (& bytes=first-last)	bytes=first-last (& None)
StackPath	bytes=first-last bytes=-suffix	bytes=first-last [& None] bytes=first-last [& None]
Tencent Cloud	bytes=first-last	None (*)

Note: F is the file size of the target resource.

TABLE II
Range Forwarding Behaviors vulnerable to OBR Attack

CDN	Vulnerable Range Format	Forwarded Range Format
CDN77	bytes=$start_1$-,$start_2$-,...,$start_n$- ($start_1 \geq 1024$)	Unchanged
CDNsun	bytes=$start_1$-,$start_2$-,...,$start_n$- ($start_1 \geq 1$)	Unchanged
Cloudflare	bytes=$start_1$-,$start_2$-,...,$start_n$-	Unchanged (*)
StackPath	bytes=$start_1$-,$start_2$-,...,$start_n$-	Unchanged [& None]

3) **CloudFront** completely adopts the *Expansion* policy to handle the `Range` header. For a `Range` header like "Range: bytes=first-last", CloudFront will replace it with "Range:

TABLE III
RANGE REPLYING BEHAVIORS VULNERABLE TO OBR ATTACK

CDN	Vulnerable Ranges Format	Response Format
Akamai	bytes=start$_1$-,start$_2$-,...,start$_n$-	n-part response (overlapping)
Azure	bytes=start$_1$-,start$_2$-,...,start$_n$- ($n \leq$64)	n-part response (overlapping)
StackPath	bytes=start$_1$-,start$_2$-,...,start$_n$-	n-part response (overlapping)

bytes=first′-last′", where first′ = (first $\gg$ 20) $\ll$ 20 and last′ = ((last $\gg$ 20 + 1) $\ll$ 20) - 1. For a `Range` header with multiple ranges like "Range: bytes=first$_1$-last$_1$,...,first$_n$-last$_n$", CloudFront will replace it with "Range: bytes=first′-last′" if last′ - first′ + 1 $\leq$ 10485760, where first′ = (min(first_list) $\gg$ 20) $\ll$ 20, last′ = ((max(last_list) $\gg$ 20 + 1) $\ll$ 20) - 1, first_list = [first$_1$, ..., first$_n$] and last_list = [last$_1$, ..., last$_n$]. For example, when receiving a `Range` header like "Range: bytes=0-0,9437184-9437184", CloudFront will change it to "Range: bytes=0-10485759". In this case, response traffic in the *client-cdn* connection is just hundreds of bytes, but the one in the *cdn-origin* connection is over 10MB.

4) **KeyCDN** adopts the *Laziness* policy for a `Range` header like "Range: bytes=first-last" and does not cache the response from the origin server. But if KeyCDN receives the same range request again, it will adopt the *Deletion* policy to handle the `Range` header. Therefore, we can send a range request twice to abuse KeyCDN to launch an SBR attack.

5) **StackPath** first adopts the *Laziness* policy to forward range requests. If it receives a 206 response from the origin server, it will continue to remove the `Range` header and then forward the new request to the origin server again. Evidently, StackPath is also vulnerable to the SBR attack.

Table II shows that 4 CDNs can be abused as the FCDN and are vulnerable to the OBR attack, including CDN7, CDNsun, Cloudflare and StackPath. The second column lists the vulnerable range formats, and the third column tells the CDNs' actual policies of handling the corresponding `Range` headers. The entries with (*) are conditional. As described above, Cloudflare allows customizing caching rules. But instead, it is vulnerable only when the target path is configured to be *Bypass*.

Table III shows that 3 CDNs can be abused as the BCDN and are vulnerable to the OBR attack, including Akamai, Azure, and StackPath. The second column lists the vulnerable multi-range formats, and the third column tells how CDNs respond to the corresponding multi-range request.

B. The Amplification Factor of the SBR Attack

As shown in Section V-A, there are 13 CDNs vulnerable to the SBR attack. We further conduct the second experiment in the wild to explore the amplification factor of this attack.

In our second experiment, the exploited `Range` header cases, listed in the second column of Table IV, are generated based on Table I. They tend to make the client receive as little traffic as possible and make the server send as much traffic as possible. As described in Section IV-B, the amplification factor is almost proportional to the target resource size. Therefore, we request different target resources ranging from 1MB to 25MB stepped by 1MB. We capture all response traffic in the *cdn-origin* connection and the *client-cdn* connection and calculate the amplification factors. The result is shown in Fig 6a-6c. And the specific amplification factors are listed in column 3-5 of Table IV when the target resource size is 1MB, 10MB, and 25MB.

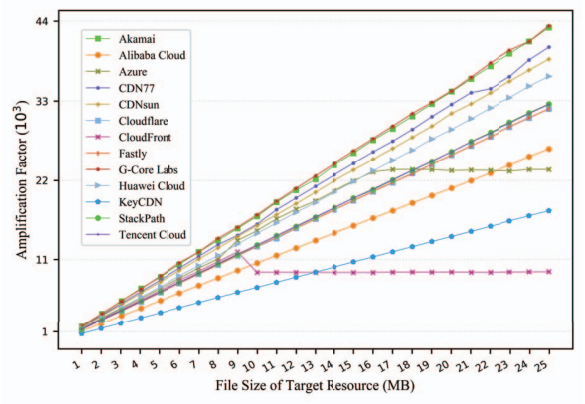
TABLE IV
THE AMPLIFICATION FACTOR VARIES WITH THE FILE SIZE OF THE TARGET RESOURCE IN THE SBR ATTACK.

CDN	Exploited Range Case	Amplification Factor		
		1MB	*10MB*	*25MB*
Akamai	bytes=0-0	1707	16991	43093
Alibaba Cloud	bytes=-1	1056	10498	26241
Azure	bytes=0-0 (F$\leq$8MB) bytes=8388608-8388608 (F$>$8MB)	1401	15016	23481
CDN77	bytes=0-0	1612	15915	40390
CDNsun	bytes=0-0	1578	15705	38730
Cloudflare	bytes=0-0	1282	12791	31836
CloudFront	bytes=0-0,9437184-9437184	1356	9214	9281
Fastly	bytes=0-0	1286	12836	31820
G-Core Labs	bytes=0-0	1763	17197	43330
Huawei Cloud	bytes=-1 (F$<$10MB) bytes=0-0 (F$\geq$10MB)	1465	14631	36335
KeyCDN	bytes=0-0 & bytes=0-0	724	7117	17744
StackPath	bytes=0-0	1297	13007	32491
Tencent Cloud	bytes=0-0	1308	12997	32438

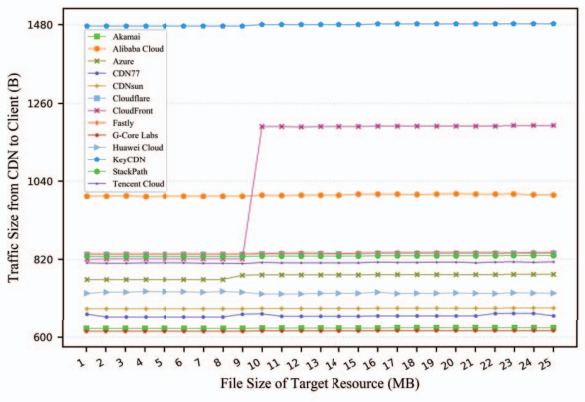
Note: F is the file size of the target resource.

As illustrated in Fig 6a-6c, response traffic in *client-cdn* connection is no more than 1500 bytes, while the amplification factor is basically proportional to the target resource size for each CDN. When the target resource size is fixed, the response traffic from the server to different CDNs is almost the same. But due to the great difference resulted from different response headers inserted by CDNs, the slope of the amplification factor varying with the target resource size is quite different. For instance, Akamai and G-Core Labs insert fewer headers to the response, causing their amplification factors to be larger than other CDNs. There are three exceptions. The first one is Azure. The response traffic from the server to Azure is up to about 16MB with the exploited Range case. When the target resource exceeds 16MB, the amplification factor of Azure will stay unchanged(Fig 6a). The second one is CloudFront. Similar to Azure, when the target resource exceeds 10MB, the amplification factor of CloudFront no longer increases(Fig 6a). The last one is KeyCDN. We need to send range requests twice each time to make a traffic amplification. Therefore, KeyCDN generates the largest response traffic(Fig 6b).

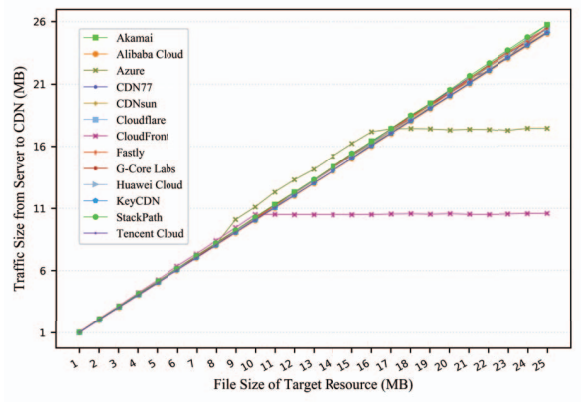
Take CloudFront as an example, when the target resource is 1MB, the amplification factor is 1356; and when the target resource exceeds 10MB, the amplification factor is about 9200. Take Akamai as another example, when the target resource is



(a) Amplification factors



(b) Response traffic from the CDN to the client



(c) Response traffic from the origin server to the CDN

Fig. 6. Exploring the amplification factor of the SBR attack with different target resources and different CDNs

1MB, the amplification factor is 1707, and when the target resource is 25MB, the amplification factor is 43093. The detail amplification factors of each CDN are listed in Table IV.

C. The Amplification Factor of the OBR Attack

As shown in Table II and Table III, 4 CDNs can be abused as the FCDN and 3 CDNs can be abused as the BCDN. Therefore, excluding the case where a CDN is cascaded with itself, there are 11 combinations of cascaded CDNs potentially vulnerable to the OBR attack, listed in column 1-2 of Table V. To find the max amplification factor, we conduct the third experiment.

In our third experiment, the test cases of multiple byte ranges, listed in the third column of Table V, are generated based on Table II and Table III. They tend to make the BCDN return as much traffic as possible. For convenience, we recognize the number of overlapping ranges as n. As described in Section IV-C, the bigger n, the larger the amplification factor.

While n is limited by the CDN's constraints on the request header size. Some CDNs even precisely restrict the number of ranges in a multi-range request. We tested the default request header size limits of related CDNs. Akamai limits the total size of all request headers to 32KB, and StackPath limits it to about 81KB. Both CDN77 and CDNsun limit the size of a single request header to 16 KB. And Cloudflare's constraints on the `Range` header can be summarized as RL + 2HHL + RHL $\leq$ 32411B, where RL is the size of the request line, HHL is the size of the `Host` header, and RHL is the size of the `Range` header. Only Azure limits the number of ranges in the `Range` header to 64. According to these results, we get the max n, as shown in the 4th column of Table V.

We use the max n to explore the max amplification factor of the OBR attack. To minimize or avoid real impacts on the performance of the corresponding vulnerable CDNs, our target resource size is limited to be just 1KB. Moreover, we set up a proxy between the FCDN and the BCDN to collect traffic transferred between them. To achieve this, we configure the FCDN's origin server as our proxy server and set the proxy server to forward requests to the BCDN. Eventually, we capture all response traffic transmitted over the *bcdn-origin* connection and the *fcdn-bcdn* connection, and calculated amplification factors, listed in column 5-7 of Table V.

As illustrated in Table V, response traffic in the *bcdn-origin* connection is no more than 2000 bytes, but the one in the *fcdn-bcdn* connection is much larger. For example, when abusing CDN77 as the FCDN and Azure as the BCDN, the max amplification factor is about 53. And when abusing Cloudflare as the FCDN and Akamai as the BCDN, the max amplification factor is about 7342. The detailed results are given in Table V.

D. Practicability of the RangeAmp Attacks

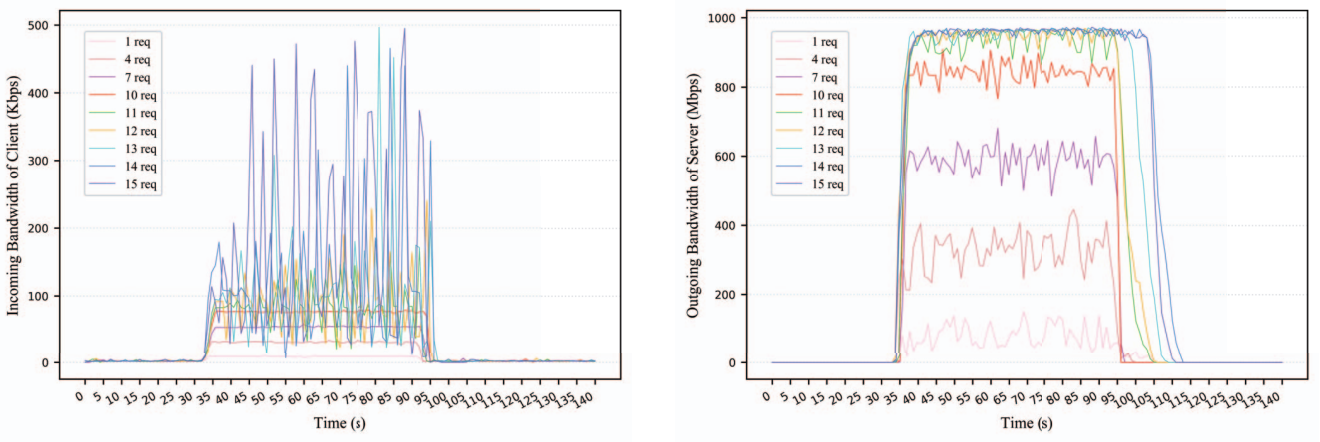
To avoid affecting the CDN's normal operation, we conduct controlled experiments in our study (see Section VI-A). But a real-world attacker can continuously and concurrently send a certain number of range requests to perform the RangeAmp attacks. In an OBR attack, the victims are specific ingress nodes of the FCDN and the BCDN. Due to an ethical concern, we can't launch a real attack to verify whether an ingress node is affected. But in an SBR attack, the victim is the origin server, thus we can evaluate the attack's impact by checking the outgoing bandwidth of our origin server.

We conduct the fourth experiment to evaluate the SBR attack's damage to bandwidth. Take Cloudflare as an example, we concurrently send m range requests to Cloudflare every

TABLE V
THE MAX AMPLIFICATION FACTOR OF THE OBR ATTACK

FCDN	BCDN	Exploited Range Case	Max n	Exploiting with 1KB of Target Resource and Max n		
				Traffic from Server to BCDN	*Traffic from BCDN to FCDN*	*Amplification Factor*
CDN77	Akamai	bytes=-1024,0-,...,0-	5455	1676B	6350944B	3789.35
	Azure		64	1620B	86745B	53.55
	StackPath		5455	1808B	6413097B	3547.07
CDNsun	Akamai	bytes=1-,0-,...,0-	5456	1676B	6337810B	3781.51
	Azure		64	1620B	84481B	52.15
	StackPath		5456	1808B	6414011B	3547.57
Cloudflare	Akamai	bytes=0-,0-,...,0-	10750	1676B	12456915B	7432.53
	Azure		64	1620B	85386B	52.71
	StackPath		10750	1940B	12636554B	6513.69
StackPath	Akamai	bytes=0-,0-,...,0-	10801	1676B	12522091B	7471.41
	Azure		64	1620B	82191B	50.74
	StackPath		-	-	-	-

Note: n is the number of overlapping ranges in the exploited multi-range request.



(a) Incoming bandwidth consumption of the client

(b) Outgoing bandwidth consumption of the origin server

Fig. 7. The bandwidth consumption of the client and the origin server with different number of attack requests

second, lasting 30 seconds. The target resource size is 10MB and the outgoing bandwidth of the origin server is 1000Mbps. During the experiment, we monitor the outgoing bandwidth of the origin server and the incoming bandwidth of the client. We iterate m from 1 to 15 to plot the trend of bandwidth consumption against time in Fig 7a and Fig 7b.

As illustrated in Fig 7a-7b, no matter how large m is, the incoming bandwidth consumption of the client is less than 500Kbps, but the outgoing bandwidth consumption of the origin server is much larger. When $m \leq 10$, it is less than 1000Mbps but almost proportional to m. When $m \geq 11$, it is close to 1000Mbps. Exactly, when $m \geq 14$, the outgoing bandwidth of the origin server is exhausted completely.

We perform the above experiment on all 13 CDNs. As expected, the experimental results are similar. Some CDNs, including Cloudflare and CloudFront, claim to have some defenses against DDoS attacks. However, during our experiments, vulnerable CDNs raised no alert while using their default configuration for the potential defenses.

E. Severity Assessment

A serious and common practical impact. According to our experiment results, the amplification factor of an SBR attack is almost proportional to the target resource size, and the one of an OBR attack is almost proportional to the number of overlapping byte ranges. All 13 CDNs we tested are vulnerable to the SBR attack, and 11 combinations of cascaded CDNs are vulnerable to the OBR attack. As we described in Section III-A, these CDNs are popular around the world and rank high in the market share. Thus, there are lots of websites and CDN nodes exposed to our RangeAmp vulnerability.

A low-cost and efficient DDoS attack. Unlike other DDoS attacks that need to control a large scale of botnets, the attacker only needs an ordinary laptop to launch the RangeAmp

attacks. The ingress nodes of CDNs are scattered around the world, coming into a natural distributed 'botnet'. This makes a RangeAmp attacker able to easily congest the target network and even create a denial of service in seconds, while the attacker pays a small cost.

A great monetary loss to the victims. Most CDNs charge their website customers by traffic consumption, including Akamai, Alibaba Cloud, Azure, CDN77, CDNsun, CloudFront, Fastly, Huawei Cloud, KeyCDN, Tencent Cloud [17]–[21]. When a website is hosted on a vulnerable CDN, its opponent can abuse the CDN to perform a RangeAmp attack against it, causing a very high CDN service fee to the website.

A security challenge to anti-DDoS. Traditional DDoS attacks that consume bandwidth mainly target the victim's incoming bandwidth. Instead, The RangeAmp attacks mainly consume the victim's outgoing bandwidth. This will pose a security challenge to the detection of DDoS attacks. As shown in Section V-D, when we abuse a CDN to perform an SBR attack, the vulnerable CDN raises no alert under its default configuration.

VI. Discussion

In this section, we will further discuss the ethics of our experiments, the root cause of RangeAmp vulnerabilities, and the mitigation solutions.

A. Ethic Consideration

When conducting real-world experiments to validating and evaluating the RangeAmp attacks, our primary concern is that when our experiments consume too much bandwidth, it may degrade the CDN's network performance and cause collateral damage to other CDN-hosted websites. Thus, we have considered this ethical concern from the beginning.

First, we conduct controlled experiments to limit bandwidth consumption in both time and volume dimensions. In the 1st and 2nd experiments, we only send one range request to the CDN each time, which hardly affects the CDN's performance. In the 3rd experiment, our target resource size is just 1KB, which will not generate excessive traffic in the *fcdn-bcdn* connection after being enlarged. In the 4th experiment, we send all requests to completely different ingress nodes of the CDN to minimize or avoid real impacts on the performance of specific nodes. And we sustain our experiment for only 30 seconds each time to keep the bandwidth consumption as little as possible.

Second, in our responsible disclosure, we unveiled our experiment details and vulnerability reproduction to the corresponding CDNs. They responded positively and are in the progress of reviewing and fixing the threats. Besides, we also contacted the editors of RFC7233, and they advised us to discuss the RangeAmp threats in the mail list of the HTTP working group. We hope that our work contributes to the security improvement of HTTP.

In summary, we make our best effort to achieve a balance between the real-world severity evaluation and the risk of impacting CDNs. And we believe our work's beneficence outweighs the damage we cause.

B. Root Cause Analysis

The range request mechanism is defined in RFC2616 [7]. This specification states that "HTTP/1.1 origin servers and intermediate caches ought to support byte ranges when possible". It explicitly specifies that if a proxy supporting range requests receives a range request, forwards the request to an inbound server, and receives an entire entity, it should only return the requested range to its client. And this is the only description related to a CDN environment.

RFC2616 is updated and published as several new RFCs (RFC7230-7239) in 2014, and the range request mechanism is specifically defined in RFC7233 [2]. Involving a CDN environment, RFC7233 only states that "origin servers and intermediate caches ought to support byte ranges when possible". Besides, RFC7233 adds some security considerations for multi-range requests, suggesting the server to ignore, coalesce or reject range requests with more than two overlapping ranges or many small ranges in the `Range` header.

However, there are no additional illustrations on range requests in the newest HTTP/2 protocol [8], which just cites the definition in HTTP/1.1, "the specification and requirements of HTTP/1.1 Range Requests [RFC7233] are applicable to HTTP/2". And we find that the RangeAmp threats in HTTP/1.1 are also applicable to HTTP/2.

As described above, RFC2616 has no security considerations for the range request mechanism. It has no restrictions on multi-range requests and even explicitly allows inconsistent response sizes between the front-end and the back-end connections of a proxy. RFC7233 realizes that the range-introduced efficiency could also bring DoS attacks against the server and gives some suggestions on multi-range requests. However, it does not clearly define how CDNs should handle a `Range` header. Even worse, RFC7540 fully refers to the definition of range requests in HTTP/1.1, without any other illustration. As a result, each CDN has its own implementation on how to handle range requests, leading to the SBR attack. Moreover, RFC7233 has already warned about the threat caused by overlapping byte ranges but some CDNs ignore it, causing the OBR attack.

Root cause: In summary, we think that the unclear definition and security negligence of the specifications constitute the root cause of RangeAmp vulnerabilities, and the implementation flaws of CDNs greatly worsen it.

C. Mitigation

Server side: Enforce local DoS defense. After deploying a CDN, customer websites are under the well-advertised DDoS protections of the CDN. However, our RangeAmp attacks can nullify this kind of protection. When suffering a RangeAmp attack, the origin server can deploy a local DoS defense (e.g. filtering requests, limiting bandwidth, etc) for temporary mitigation. But this does not necessarily work. From the perspective of the origin server, attack requests are no different from benign requests and come from widely distributed CDN nodes. It is difficult for the origin server to defend against it effectively without affecting normal services.

CDN side: Modify the specific implementation on range requests. CDNs can detect and intercept malicious range requests based on the characteristics of the RangeAmp attacks. But the essential approach is to improve the policy of handling the `Range` header. As described in Section IV-B, the *Deletion* policy and the *Expansion* policy cause the SBR attack. Therefore, CDNs can adopt the *Laziness* policy to completely defend against the SBR attack. But this also makes CDNs unable to benefit from range requests. A better way is to adopt the *Expansion* policy but not extend the byte range too much. For example, it is acceptable to increase the byte range by 8KB, which will not cause too much traffic difference between the CDN's front-end and back-end connections. In addition, CDNs should follow the security recommendations on multi-range requests in RFC7233, such as rejecting range requests with many small ranges or multiple overlapping ranges in the `Range` header. Furthermore, as an important part of the Internet infrastructure, we believe CDNs should perform a full security evaluation before supporting new protocol features.

Protocol side: Revise a well-defined and security-aware RFC. As discussed in Section VI-B, the unclear definitions and insufficient security considerations of the specifications essentially cause RangeAmp threats. Thus, we contacted the editors of RFC7233 and they agreed that this kind of attack should be mentioned as a security consideration. According to their suggestions, we will continue to discuss this threat on the mailing list of the HTTP working group. We think that a more specific limit of the `Range` header should be defined in a future updated RFC, especially for the HTTP middle-boxes like CDNs.

VII. Responsible Disclosure

A. Response from CDN vendors

All vulnerabilities found in our study have been reported to related CDN vendors. We actively contacted vendors one by one more than one month before the paper was submitted. We provided them mitigation solutions to eliminate the detected threats. Most vendors quickly confirmed the vulnerabilities and claimed to fix them as soon as possible. Some vendors have indeed fixed the vulnerabilities, including CDN77, Huawei Cloud, G-Core Labs, and Tencent Cloud. Unfortunately, although we disclosed RangeAmp issues to StackPath in several ways, including the StackPath Support platform, email, and customer services, we did not receive any feedback. (Six months later, StackPath contacted us and explained that they had responded quickly to our reported RangeAmp attacks, but their mail system failed to send their feedback to us. They claimed to deploy a fix across all StackPath edge locations to mitigate the OBR attack. And they will continue to monitor and evaluate RangeAmp attacks.)

In general, we have tried our best to responsibly report the vulnerabilities and provide mitigation solutions. The related vendors will have about seven months to implement mitigation techniques before this paper is published. And they have the duty to inform their customers about the vulnerabilities.

The responses from CDN vendors are summarized below:

Cloudflare appreciated our work and discussed the vulnerabilities with us in detail. They thought that the SBR attack relies on constantly triggering a cache-miss and a customer can add a page rule to ignore query strings. But this does not solve the problem fundamentally. The malicious customers and some normal customers will not follow this suggestion. Unfortunately, they won't implement our mitigation solutions **because Cloudflare does not want to cache partial responses of certain resources**. And they insisted that they are not deviating away from the specifications. But Cloudflare now seems to have improved its DDoS detection mechanism.

Huawei Cloud evaluated the issue as a high-risk vulnerability. They viewed it as indeed a problem for the CDN industry and contacted us actively to discuss how to defend against it. And they have now fixed the related vulnerabilities.

CDN77 thanked us for our research. To defend against the OBR attack, they have created a detection for overlapping ranges and such requests will be denied. Besides, they are now moving to disable the `Range` header to mitigate the SBR attack and try implementing slicing of range requests.

G-Core Labs determined that the vulnerabilities exist on their service and contacted us actively to discuss mitigation options. To defend against the SBR attack, they eventually chose to make the "slice" option enabled by default, which adopts the *Laziness* policy to handle the `Range` header.

Tencent Cloud confirmed that their implementation is vulnerable to the SBR attack. And they have fixed it now.

Akamai acknowledged that the Azure case of Akamai is indeed problematic. They said that the Azure configurations may override the origin configurations of Akamai and this issue should be fixed on the Azure side. Anyway, they claimed to look into this problem promptly.

CloudFront admitted the methods used to optimize caching of range requests indeed increase bytes requested from the origin. They claimed to investigate how to prevent multi-range requests from increasing any more traffic than single-range requests. Besides, they claimed that they have safeguards to prevent excessive back-to-origin requests.

CDNsun claimed that they would mitigate the OBR attack by limiting the number of ranges or rejecting overlapping range requests. But they insisted that they don't have a proper technical solution to the SBR attack, although we introduced some mitigation options to them.

Fastly expressed appreciation for our study and informed us that they are investigating to validate the attack scenarios, explore the effectiveness of mitigation already available, and develop new capabilities to manage the risk of this attack.

Azure confirmed that the attack is feasible, but only in certain circumstances. They insisted that if a customer configures the options to ignore query strings then the attack will be mitigated. But as we discussed with Cloudflare, a malicious customer won't follow the security best practices. And some normal customers won't do so as well.

Alibaba Cloud confirmed that their implementation is vulnerable to the SBR attack. They are currently fixing it.

KeyCDN thanked us for our report but claimed that they have already been aware of this issue.

B. Response from the RFC editor

We also contacted the honored editors of RFC7233 with email and Roy T. Fielding replied that "it's the CDN's responsibility to manage its back-end behavior regardless of the protocols used to access it". But he also said that "I agree that this kind of attack should be mentioned as a security consideration" and "we can warn about certain effects and kinds of attack". He suggested we discuss this threat on the mailing list of the HTTP working group.

VIII. Related Work

HTTP Range Security. To the best of our knowledge, there is no academic literature discussing the security risks introduced by range requests in a CDN environment. According to the CVE platform, there are about 20 vulnerabilities related to range requests. All of them are related to improper implementations but have nothing to do with CDNs. For instance, CVE-2017-7529 [22] presents an integer overflow caused by Nginx's incorrect processing with the `Range` field. And CVE-2011-3192 [13] presents a DoS attack using a `Range` header with multiple ranges to exhaust memory on the Apache server. Our RangeAmp attacks mainly exploit the asymmetrical traffic consumption in the front-end and back-end connections of a CDN, which is quite different from these vulnerabilities.

CDN Security. As an important part of the Internet infrastructure, CDN security has been well researched. Due to the DDoS protection provided by CDNs, attackers are highly interested in finding out the origin IP of the target website. There are some methods based on information leaks to expose the sensitive information of the origin server [6], [9]. In comparison, our RangeAmp attacks can directly nullify the DDoS protection of a vulnerable CDN and abusing the CDN to attack the origin server. Triukose et al [5] proposed an attack of exhausting the bandwidth of the origin server by rapidly dropping the front-end connections. We evaluated this attack and found that most CDNs can mitigate it. They will break the corresponding back-end connections when the front-end connections are abnormally cut off. However, this defense is invalid under our RangeAmp attacks. Furthermore, CDNs can also become the victim, and in the forwarding-loop attacks proposed by Chen et al [3], an attacker can chain CDN nodes into a loop, causing the malicious request to be processed repeatedly and reducing CDN's availability. Differently, our RangeAmp attacks present a novel method to perform an amplification attack against specific ingress nodes of CDNs. Some studies show that the global distribution and massive nodes of a CDN also facilitate malicious CDN customers to abuse the CDN [23]–[25]. And CDN's mappings between clients and surrogates can also be maneuvered with crafted DNS records [26]. Compared to previous researches related to CDNs, we propose a novel class of amplification attacks and conduct a real-world security evaluation with 13 popular CDNs. Indeed, we provide a complement to existing CDN security research.

Amplification Attacks. Amplification attacks have also long been well studied. Booth et al [27] reveal that, by recruiting UDP servers on the Internet as the reflectors, a UDP amplification attack can reach an amplification factor of 556. Sieklik et al [28] further analyze the DNSSEC based amplification attack, which leads to an amplification factor of 44. Besides the UDP protocol, the TCP protocol can also be abused. Anagnostopoulos et al [29] study the TFTP amplification attack with an amplification factor of 60. Kührer et al [30] gave an in-depth analysis of the TCP reflection attack across famous TCP services, eg, HTTP, MySQL, and POP3 services. Further, Kührer et al [31] also reveals that the NTP service can lead to an amplification factor of 4670. In 2015, Krämer et al [32] designed a novel honeypot to track and analyze these types of amplification attacks. Compared with these previous amplification attack studies, our RangeAmp attack can reach a much larger amplification factor. More importantly, when the target website is hosted behind a CDN, the CDN can defend against all amplification attacks of previous studies. However, a RangeAmp attacker can nullify the DDoS protection provided by CDNs and achieve severe amplification damage against the CDN-hided website server.

In brief, our study reveals that the `Range` header can be exploited to perform a novel class of amplification attacks against the websites hosted on CDNs and the ingress nodes of CDNs. This attack nullifies the CDN-provided DDoS protections and poses a severe threat to the Internet security ecosystem.

IX. Conclusion

We have presented the principles of the RangeAmp vulnerability, along with a comprehensive study of its practicality in the wild. We find that the 13 popular CDNs tested are all vulnerable. The unclear definition and security negligence of the specifications are the root cause, and the implementation flaws of CDNs further worsen this vulnerability. We believe that the RangeAmp attacks will pose severe threats to the serviceability of CDNs and the availability of websites. We hope that our study will provide insight into this vulnerability and help the potentially relevant victims to fully understand them. In the short term, we suggest that the CDNs and websites adopt one or more of the mitigation solutions discussed in our paper. In the longer term, we think that a more specific limit of range requests should be defined in a future updated RFC, especially for the HTTP middle-boxes like CDNs.

Acknowledgement

Special thanks are expressed to our shepherd Marc Dacier and the anonymous reviewers for their insightful comments, which contributed to a great improvement of our paper. We also thank the RFC7233's editor Roy T. Fielding and all related CDN vendors for their valuable feedback. Last but not least, we gratefully acknowledge the help of our friend Wenchang Ma, who spent lots of time correcting our grammar mistakes.

REFERENCES

[1] Data Economy, "Data economy frontline. how close you need to be to edge data?" https://data-economy.com/data-economy-frontline-how-close-you-need-to-be-to-edge-data/, [Accessed Dec. 2019].

[2] R. T. Fielding, Y. Lafon, and J. F. Reschke, "Hypertext transfer protocol (HTTP/1.1): range requests," *RFC*, vol. 7233, pp. 1–25, 2014.

[3] J. Chen, X. Zheng, H. Duan, J. Liang, J. Jiang, K. Li, T. Wan, and V. Paxson, "Forwarding-loop attacks in content delivery networks," in *23rd Annual Network and Distributed System Security Symposium, NDSS 2016*. The Internet Society, 2016.

[4] J. Chen, J. Jiang, H. Duan, N. Weaver, T. Wan, and V. Paxson, "Host of troubles: Multiple host ambiguities in HTTP implementations," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1516–1527.

[5] S. Triukose, Z. Al-Qudah, and M. Rabinovich, "Content delivery networks: Protection or threat?" in *Computer Security - ESORICS 2009, 14th European Symposium on Research in Computer Security*, ser. Lecture Notes in Computer Science, vol. 5789. Springer, 2009, pp. 371–389.

[6] T. Vissers, T. van Goethem, W. Joosen, and N. Nikiforakis, "Maneuvering around clouds: Bypassing cloud-based security providers," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 1530–1541.

[7] R. T. Fielding, J. Gettys, J. C. Mogul, H. F. Nielsen, L. Masinter, P. J. Leach, and T. Berners-Lee, "Hypertext transfer protocol - HTTP/1.1," *RFC*, vol. 2616, pp. 1–176, 1999.

[8] M. Belshe, R. Peon, and M. Thomson, "Hypertext transfer protocol version 2 (HTTP/2)," *RFC*, vol. 7540, pp. 1–96, 2015.

[9] L. Jin, S. Hao, H. Wang, and C. 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*. IEEE Computer Society, 2018, pp. 362–373.

[10] ——, "Unveil the hidden presence: Characterizing the backend interface of content delivery networks," in *27th IEEE International Conference on Network Protocols, ICNP 2019*. IEEE, 2019, pp. 1–11.

[11] Datanyze, "Content delivery networks market share report," https://www.datanyze.com/market-share/cdn/, [Accessed Dec. 2019].

[12] Rapid7, "Apache range header dos (apache killer)," https://www.rapid7.com/db/modules/auxiliary/dos/http/apache_range_dos, [Accessed Dec. 2019].

[13] cve.mitre.org, "Cve-2011-3192," https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-3192, [Accessed Oct. 2019].

[14] R. Guo, W. Li, B. Liu, S. Hao, J. Zhang, H. Duan, K. Shen, J. Chen, and Y. Liu, "Cdn judo: Breaking the cdn dos protection with itself," *NDSS'20*, 2020.

[15] D. Senecal, "Slow dos on the rise," https://blogs.akamai.com/2013/09/slow-dos-on-the-rise.html, [Accessed Oct. 2018].

[16] Cloudflare, "Slowloris ddos attack," https://www.cloudflare.com/learning/ddos/ddos-low-and-slow-attack, [Accessed Oct. 2018].

[17] CDNPerf, "Cdn calculator - cdncalc - cdnperf," https://www.cdnperf.com/tools/cdn-calculator, [Accessed Dec. 2019].

[18] CDNsun, "Best cdn solutions at cheap price — cdn pricing," https://cdnsun.com/pricing, [Accessed Dec. 2019].

[19] Alibaba Cloud, "Cdn (content delivery network) pricing & purchasing methods - alibaba cloud," https://www.alibabacloud.com/product/cdn/pricing?spm=a3c0i.7938564.220486.75.26d62aecnaGFXf, [Accessed Dec. 2019].

[20] HUAWEI CLOUD, "Price calculator-huawei cloud," https://intl.huaweicloud.com/en-us/pricing/index.html?tab=detail#/cdn, [Accessed Dec. 2019].

[21] Tencent Cloud, "Pricing center - tencent cloud," https://intl.cloud.tencent.com/pricing/cdn, [Accessed Dec. 2019].

[22] M. Dounin, "Cve-2017-7529,," http://mailman.nginx.org/pipermail/nginx-announce/2017/000200.html, [Accessed Oct. 2019].

[23] J. Holowczak and A. Houmansadr, "Cachebrowser: Bypassing chinese censorship without proxies using cached content," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 70–83.

[24] H. Zolfaghari and A. Houmansadr, "Practical censorship evasion leveraging content delivery networks," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 1715–1726.

[25] R. Guo, J. Chen, B. Liu, J. Zhang, C. Zhang, H. Duan, T. Wan, J. Jiang, S. Hao, and Y. Jia, "Abusing cdns for fun and profit: Security issues in cdns' origin validation," in *37th IEEE Symposium on Reliable Distributed Systems, SRDS 2018*. IEEE Computer Society, 2018, pp. 1–10.

[26] S. Hao, Y. Zhang, H. Wang, and A. Stavrou, "End-users get maneuvered: Empirical analysis of redirection hijacking in content delivery networks," in *27th USENIX Security Symposium, USENIX Security 2018*. USENIX Association, 2018, pp. 1129–1145.

[27] T. G. Booth and K. Andersson, "Elimination of dos UDP reflection amplification bandwidth attacks, protecting TCP services," in *Future Network Systems and Security - First International Conference*, ser. Communications in Computer and Information Science, vol. 523. Springer, 2015, pp. 1–15.

[28] B. Sieklik, R. Macfarlane, and W. J. Buchanan, "Evaluation of TFTP ddos amplification attack," *Comput. Secur.*, vol. 57, pp. 67–92, 2016.

[29] M. Anagnostopoulos, G. Kambourakis, P. Kopanos, G. Louloudakis, and S. Gritzalis, "DNS amplification attack revisited," *Comput. Secur.*, vol. 39, pp. 475–485, 2013.

[30] M. Kührer, T. Hupperich, C. Rossow, and T. Holz, "Hell of a handshake: Abusing TCP for reflective amplification ddos attacks," in *8th USENIX Workshop on Offensive Technologies, WOOT '14*. USENIX Association, 2014.

[31] ——, "Exit from hell? reducing the impact of amplification ddos attacks," in *Proceedings of the 23rd USENIX Security Symposium*. USENIX Association, 2014, pp. 111–125.

[32] L. Krämer, J. Krupp, D. Makita, T. Nishizoe, T. Koide, K. Yoshioka, and C. Rossow, "Amppot: Monitoring and defending against amplification ddos attacks," in *Research in Attacks, Intrusions, and Defenses - 18th International Symposium, RAID 2015*, ser. Lecture Notes in Computer Science, vol. 9404. Springer, 2015, pp. 615–636.