Abusing CDNs for Fun and Profit: Security Issues in CDNs' Origin Validation

Run Guo*, Jianjun Chen*, Baojun Liu*, Jia Zhang* (✉), Chao Zhang* (✉),
Haixin Duan* ‖ (✉), Tao Wan†, Jian Jiang‡, Shuang Hao§ and Yaoqi Jia¶
*Tsinghua University, †Huawei Canada, ‡Shape Security, §University of Texas, Dallas, ¶Zilliqa Research
‖Tsinghua University-360 Enterprise Security Group Joint Research Center

***Abstract*—Content Delivery Networks (CDNs) are critical Internet infrastructure. Besides high availability and high performance, CDNs also provide security services such as anti-DoS and Web Application Firewalls to CDN-powered websites. However, the massive resources of CDNs may also be leveraged by attackers exploiting their architectural, implementation, or operational weaknesses.**

In this paper, we show that today's CDN operation is overly loose in customer-controlled forwarding policy and the lack of origin validation leads to a wide range of abuse cases such as DoS attack and stealthy port scan. We systematically study these abuse cases and demonstrate their feasibility in popular CDNs. Further, we evaluate the impact of these abuses by discovering that there are millions of CDN edge servers, and a substantial fraction of them can be abused. Lastly, we propose mitigation solutions against such abuses and discuss their feasibility.

I. Introduction

A Content Delivery Network (CDN) usually deploys edge servers or surrogates in different geographical locations that are often across the global Internet backbone, working as a distributed network service with a significant capacity of both computational resources and network bandwidth. By caching contents for its customers' websites, a CDN redirects web requests from end-users to geographically nearby surrogates, providing websites with both high global availability and improved network performance. Due to a wide range of benefits offered by CDNs, numerous customers spanning from small to large websites are now deployed behind CDNs. As a result, CDNs have become a critical Internet infrastructure, and brought billions of annual revenue to their providers [1].

In order to compete in this booming market, most CDNs offer potential customers *free* or *free-trial* services, as well as flexible configuration options, such as HTTP header manipulation, to minimize deployment efforts and required changes in customers' websites. However, over-emphasis of convenience also brings security flaws.

In this paper, we investigate the security of CDNs, and point out CDNs could be abused in many ways. The root cause is that, CDNs do not validate the customer-supplied website *origin* option properly. The *origin* is a domain name or an IP address that a CDN will forward requests to and pull resources from, an ordinary CDN customer normally configures the origin to point to the web server he owns. However, as CDNs don't validate the customer's ownership of the *origin*, a malicious CDN customer could drive CDN surrogates launching TCP requests to an arbitrary domain name or IP address. Accompanied by other flexible configurations provided by CDNs, attackers could abuse CDNs to access third party resources in unintended ways. For example, an abuser can utilize surrogates as proxies to access resources restricted per IP, to circumvent Internet censorship, to scan TCP ports covertly and to launch DoS attack etc. The fact that CDNs provide *free* or *free-trial* services further reduces the cost of launching such abuses.

It is worth noting that, different CDNs provide varied configuration options, and certain CDNs have enforced restrictions on customers' configurations. For example, some CDNs have a white-list of TCP ports that surrogates could connect to. These restrictions could invalidate some abuses aforementioned, e.g., stealthy TCP port scan. However, we point out that surrogates from different CDNs could be chained together to bypass such restrictions. By systematically studying features provided by eight different CDNs, we find out six possible abuse cases.

Furthermore, we have evaluated the impacts of such abuses. We found out millions of globally distributed surrogates could be abused. Attackers could simply register a CDN service, and then build a controllable *proxy pool* using these surrogates. Even worse, since CDNs have become an indispensable part of the Internet and served most of the popular websites, when third parties are under attack due to CDN abuses, it is impossible to stop the attack by simply blocking CDNs' IP addresses without causing collateral damages.

In summary, we make the following contributions:

1) We point out the *CDN Origin Abuse* that a malicious CDN customer can abuse CDNs as the proxy to access any website or establish a TCP connection to any TCP port, due to the fact that CDNs don't validate the customer-supplied origin. Accompanied with CDNs' rich configuration options, this *Origin Abuse* opens the door for a wide range of abuses.
2) By studying features provided by 8 popular CDNs, we specifically outline 3 categories of CDN abuses, which rely on CDNs' three features respectively, i.e., flexible configurations, network proxy, and global Geo-IP distribution. And we systematically design 6 abuse cases to evaluate the applicability of such abuses on the Internet.
3) We evaluate the impact of these abuses by measuring the volume of CDN surrogates. We propose methods to find out surrogates' distribution, and have measured the numbers of CDNs' user-facing IPs and website-facing IPs. We find millions of surrogates that are globally distributed. The abundance and the global Geo-distribution of these surrogates greatly amplify the severity of CDN abuses.

We agree that some attack techniques (e.g., bypassing IP Geo-blocking) discussed in this paper aren't new in themselves. But the central theme of this work is that these attacks can be conducted by exploiting CDN vulnerabilities or features. The contribution of this work is to show that CDN as a critical Internet service, has enabled various abuses or attacks, due to both of its lack of security consideration and inherent difficulties as a large-scale proxy network with weak authorization. Nevertheless, some of those CDN-based attacks may not cause more severe damages than those not abusing CDNs. However, some other attacks (e.g., millions of high bandwidth CDN nodes can be abused in scanning and DoS.) can do impose a serious threat.

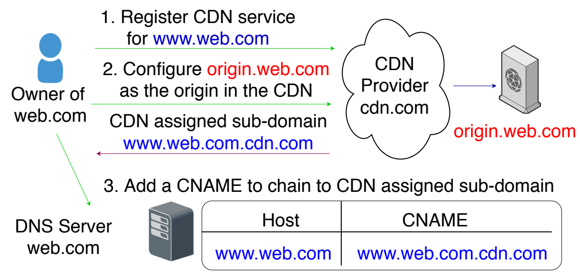
We hope the discussions of a wide range of abuse cases help serve the purpose of raising security awareness among CDN providers and motivating them to fix the problems, resulting in improved security of critical Internet infrastructure.

II. Background

As a distributed Internet infrastructure, a CDN offloads network traffic from a website origin by caching its resources, and a CDN also speeds up end-users' accessing delay by deploy a large number of CDN surrogates with high bandwidth around the world.

From an end user's view, when he or she tries to access a CDN-powered website, CDN's request-routing mechanism [2] will redirect HTTP request to a surrogate that could best serve the request. For example, in "DNS rerouting" mechanism, the website domain name is first resolved to a CDN assigned subdomain. Then, the domain name system of a CDN network is responsible for returning the surrogate IP that could best serve the request based on metrics such as network proximity, bandwidth availability. After this DNS resolving process, requests are sent to the surrogate IP returned in DNS response. The chosen surrogate primarily inspects the "Host" header and URL within the incoming requests, and it decides whether to serve the requests with locally cached contents or fetch the requested contents from the website origin associated with that "Host" header.

Although CDNs use the request-routing mechanism to schedule surrogates, an end user can also bypass it by directly sending HTTP requests to a surrogate's IP address. In these requests, the "Host" header is filled with the CDN-powered website's domain name, to notify the surrogate of the website he or she wants to access [3], [4].



1. Register CDN service for www.web.com
2. Configure origin.web.com as the origin in the CDN
CDN assigned sub-domain www.web.com.cdn.com
Owner of web.com
CDN Provider cdn.com
origin.web.com
3. Add a CNAME to chain to CDN assigned sub-domain
DNS Server web.com
Host
CNAME
www.web.com
www.web.com.cdn.com

Fig. 1: *Procedures to deploy a website behind a CDN*

From a website owner's view, in order to deploy a website behind a CDN and make "DNS rerouting" work, the owner can follow procedures as simply illustrated in Fig 1. Firstly, the website owner has to register a CDN's service to become a customer. Secondly, the website owner should configure options in the CDN's customer interface, such as caching and forwarding policies. Among these options, the website origin is a required field, it directs the CDN from where to fetch the requested resources back. Thirdly, the website owner use CNAME to chain to the CDN-assigned subdomain. Thus, the domain name system of a CDN network is empowered to resolve the DNS query and return the surrogate IP.

As we can see, the website origin is a critical option, and it's directly configured by the CDN customer. However, for 8 CDNs we examined, we find such customer-configured option lacks proper validation. In the next section, we'll analyze this origin validation problem in detail and present various abuses built on it.

III. Categories of CDN Abuse

A. *The Root Cause Analysis*

As illustrated in Section II, CDNs will forward cache-missed requests to the website origin. To specify where the website origin is located, CDNs all provide an origin option in their customer interface. As a CDN customer, the website owner can configure the origin address either in an IP address or domain name format, as shown in Table I.

However, our experiments show that most CDNs do not validate such customer-supplied critical information properly (see Table I). For example, among the 8 leading CDNs, 3 of them validate neither the IP nor domain name configured in the origin option, 3 of them validate origin IPs by filtering out private IPs. Cloudflare filters already-registered customers' domains, selected well-known domains (e.g., Facebook and Twitter), and some public domains that provide sub-domain service (e.g., GitHub). None of these CDNs validate whether their customers own the configured IPs or domain names.

CDN Origin Abuse: Due to the lack of proper origin validation, an abuser can register to be a CDN customer and configure the origin to be the nearly arbitrary domain or IP not owned by the abuser. Then, the CDN will forward any subsequent cache-missed requests to this configured origin. Besides, empowered with request-routing bypassing mechanism, an abuser can choose which surrogates to be abused to forward requests. Therefore, a malicious CDN customer can abuse a CDN as a proxy pool to forward requests back to nearly arbitrary origin, and the abuser can directly schedule which and how many proxies to be abused. We name such abuse as the *CDN Origin Abuse*, and it builds the foundation for a wide range of abuses.

Knowing that origin can be configured arbitrarily by a CDN customer, we systematically evaluate how CDNs can be abused in each of those CDN distinct features. We outline 6 abuse cases (see Table II) and classify them into 3 categories namely *CDN Option Abuse*, *CDN Proxy Abuse*, and *CDN Geo-IP Abuse*. We acknowledge that our abuse cases are likely incomplete and other cases may exist. Nevertheless, they serve the purpose of demonstrating that CDNs are subject to extensive abuses and require significant security enhancement.

TABLE I: *CDNs' Origin Option and Validation Behavior*

	Configure IP	IP Verification Behavior	Configure Domain Name	Domain Verification Behavior
Akamai	✓	No Verification	✓	No Verification
Amazon	N/A	N/A	✓	No Verification
Azure	✓	No Verification	✓	No Verification
Baidu	✓	Private IP Blacklist	✓	No Verification
Cloudflare	✓	Private IP Blacklist	✓	Domain Blacklist
Fastly	✓	No Verification	✓	No Verification
Incapsula	✓	Private IP Blacklist	N/A (Free-Trial)	N/A
MaxCDN	N/A	N/A	✓	No Verification

TABLE II: *The categories of CDN Abuse*

Abuse Category	Abuse Cases	Experiments
CDN Option Abuse	HTTP Header Manipulation	Exp.IV-A: Chain CDNs together with "Host" header modification
		Exp.IV-B: Reset HTTP headers to avoid IP-based user tracking
	Origin Port Abuse	Exp.IV-C: CDN-proxied TCP port scanning
		Exp.IV-D: TCP concurrent connection exhaustion attack
CDN Proxy Abuse	Forwarding Abuse	Exp.IV-E: Point the origin to censored websites to circumvent censorship
	Segmented TCP Abuse	Exp.IV-F: CDN-based DoS attack against any websites
CDN Geo-IP Abuse	IP Abundance Abuse	Exp.IV-G: Internet survey fraud through CDNs
	IP Geo-distribution Abuse	Exp.IV-H: Bypass web service's IP Geo-blocking

B. CDN Option Abuse

In order to minimize customers' deployment efforts, CDNs provide their customers with rich configuration options to control caching and forwarding behaviors. However, insufficient validation of these options can also allow a malicious CDN customer to abuse CDNs. Two cases of abusing CDN configuration options are discussed below.

1) HTTP Header Manipulation: CDNs commonly allow customers to configure how HTTP headers in requests should be modified when being forwarded by CDNs. For example, CDNs allow the change of "Host" header, a critical field in determining how a request is routed, e.g., to which domain on a shared web server. Other HTTP headers can also be modified too, such as "X-Forwarded-For", we list these modifiable headers in Table IV of Section IV-B.

Two abuse cases are designed to demonstrate the HTTP header manipulation, including Exp.IV-A that modifies "Host" header to chain CDNs together to enlarge the attack surface of after-mentioned abuses, and Exp.IV-B that resets HTTP headers to avoid user tracking.

2) Origin Port Abuse: Besides IP and domain name, some CDNs also allow their customers to configure the port number of the website origin. An attacker can abuse this option to forward HTTP requests to arbitrary TCP port, regardless of the state of this port or what services are running on it.

We find that, when encountering errors, CDNs' error responses could expose the port status of the website origin. With such unintended information disclosure, CDNs can be abused for a stealthy port scan, since it is the CDN's IP, not abuser's own IP, that is logged by the target. In Exp.IV-C, we validate the possibility of a CDN-proxied scan by examining different CDNs' error messages.

Even when a non-HTTP TCP service is running on the port of website origin, surrogates can still complete TCP's 3-way handshakes and establish a connection with the website origin, bypassing the SYN cookies [5] defense. In Exp.IV-D, we abuse a CDN to establish TCP connections with our testing DNS server's TCP port 53, resulting in DoS effect by exceeding the server's TCP concurrent connection limit.

C. CDN Proxy Abuse

A CDN works as the traffic proxy between end users and the website origin, it divides an end-to-end TCP connection into two segments (or more when multiple CDN-internal nodes are involved). We consider 2 abuse ways here.

1) Forwarding Abuse: With the *origin abuse*, a CDN customer is able to configure the origin server to be a sensitive website which might be inaccessible for local network environment. In Exp.IV-E, we show that an abuser can circumvent Internet censorship by configuring a censored website as the origin, and get restricted contents via a CDN network.

2) Segmented TCP Abuse: The two TCP segments (namely front-end and back-end respectively) created by the CDN can have inconsistent TCP states. For example, a back-end segment may still be in transmission while the associated front-end segment has closed connection. As noted in [3], these inconsistent TCP states can be abused, e.g., to launch a DoS attack against a CDN-powered website. We re-evaluated this abuse case (cf IV-F) and discovered that certain mechanisms (e.g., abort the back-end connection when the front-end connection closes) have been deployed by CDNs to mitigate this type of abuse. However, we also found that this mitigation can be defeated with the *Sockstress* [6] trick of using small TCP window to read slowly in the front-end connection. Further, with the *origin abuse*, the severity of this CDN-based DoS attack is amplified by pointing the origin to any websites.

D. Geo-IP Abuse

The characteristic of widely Geo-distribution of CDN server could empower an abuser to forward requests to a website

TABLE III: *CDNs' Host Modification Behavior*

	Request Domain	Origin Domain	Any Domain
Akamai	✓	✓	
Amazon		✓	
Azure	✓	✓	✓
Baidu	✓		
Cloudflare	✓	N/A(Free Plan)	N/A(Free Plan)
Fastly	✓	✓	✓
Incapsula	✓		
MaxCDN	✓	✓	✓

from different origin points. And the massive CDN surrogates allows following two kinds of abuse.

1) IP Abundance Abuse: As CDNs have deployed a large number of surrogates, the IP abundance of these surrogates can be abused to bypass restriction per IP address. For example, Internet survey website "polldaddy.com" records voter's IP to enforce the restriction of one vote per IP, as shown in Exp.IV-G. However, we successfully voted hundreds of times through different CDN surrogates.

The IP abundance can also amplify the severity of other CDN abuses as well, e.g., leveraging different IPs, a CDN-proxied scan can be speed-ed up, or a CDN-based DoS attack can be launched in parallel.

2) IP Geo-Distribution Abuse: As CDNs endeavor to deploy surrogates globally, such massive Geo-distribution can be used to bypass the restriction on IP's Geo-location. In Exp.IV-H, we demonstrate that the Geo-blocking enforced by the music website "www.pandora.com" can be easily bypassed.

IV. Practical Experiments of CDN Abuse

We performed extensive experiments to validate the abuse cases outlined in Section III, and we report our technical detail here.

A. Chain CDNs together with "Host" header modification

When a CDN surrogate forwards requests, the surrogate can either keep the "Host" header unchanged, or modify it to the configured origin's domain name.

As in Table III, 5 out of 8 CDNs support modifying the "Host" header to the origin domain. And Azure, Fastly and MaxCDN even support modifying the "Host" header to any domain. Thus, with CDNs' no origin validation behavior, an abuser can configure CDNs to forward HTTP requests to any target website's domain name or IP, and these requests will be processed normally as the modified "Host" header correctly matches to the target website's domain name.

Besides, with "Host" header modification ability, an attacker can add CNAME records as in Figure 2, to chain two CDNs together. It is also possible to chain more CDNs to create a request-forwarding line, requests sent from the attacker can sequentially be forwarded back to the final origin.

We succeeded in accessing our website by sequentially chaining Microsoft Azure and Amazon CloudFront, as shown in the "Via" and "X-Forwarded-For" headers of HTTP requests forwarded to the website. We can see that, firstly Azure adds its distinctive header value "HTTP/1.1 ECCAcc (hhp/9AB2)"

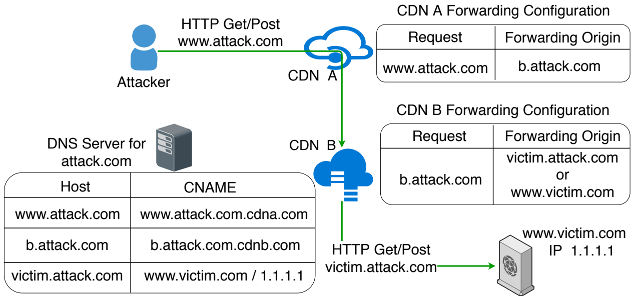


Fig. 2: *Chain CDNs together to create a request-forwarding line*

in "via" header, together with an Azure IP in "X-Forwarded-For" header, then CloudFront appends "*.cloudfront.net" and a CloudFront IP.

```
via: HTTP/1.1 ECCAcc (hhp/9AB2),
    3f218bd55e2fcd2fab19328ef2d398bc.cloudfront.net
X-Forwarded-For: azure.ip, cloudfront.ip
```

This chaining is particularly useful in enlarging the attack surface of aforementioned CDN abuses, e.g., to make CDN abuses to be more difficult to trace back when different CDNs' surrogates are abused as the proxy chain, or to bypass port limitation (See Exp. IV-C) or forwarding restriction (See Exp. IV-F) enforced by some CDNs.

B. Reset HTTP headers to avoid IP-based user tracking

Although browser's "incognito mode" keeps the end-user's privacy by not exposing any sensitive information, "incognito mode" can't hide the end-user's IP from user-tracking websites. Thus, in order to avoid being tracked by websites, besides browser's "incognito mode", the end-user may additionally use a public proxy to hide the IP address. However, the public proxy could automatically add HTTP headers such as 'x-forwarded-for', 'HTTP_X_FORWARDED_FOR' or 'Via' when forwarding requests [7], which still expose the end-user's real IP to websites.

Different from the public proxy, CDNs support filtering HTTP headers when forwarding requests, and a CDN customer can directly control this filtering behavior. Thus, when an abuser configures the target website as the origin, CDNs can both filter HTTP headers and hide the end-user's IP address. When browsing the website through CDNs, CDNs work as the "anonymization proxy" to not expose any user-related information to the website.

In experiments, we examine whether CDNs will pass any user-related information back to the origin, and whether the information can be reset or filtered. The results are shown in Table IV, e.g., CloudFront by default pass four user-related headers back to the origin, but CloudFront also support filtering these headers.

Although 5 out of 8 tested CDNs don't support filtering any headers, with the CDN-chaining trick to chain more than two CDNs together, this abuse is still applicable when one of CDNs supports header manipulation,

TABLE IV: *HTTP Header Modification Behavior*

	Headers that can expose end-user identity	Reset/Filter
Akamai	HTTP_X_FORWARDED_FOR, X-Forwarded-For, User-Agent, Cookie	N/A
Amazon CloudFront	HTTP_X_FORWARDED_FOR, X-Forwarded-For, User-Agent, Cookie	√
Azure	HTTP_X_FORWARDED_FOR, X-Forwarded-For, User-Agent, Cookie	N/A
Baidu	HTTP_X_FORWARDED_FOR, Cf-Connecting-Ip, X-Forwarded-For, User-Agent, Cookie	N/A
Cloudflare	HTTP_X_FORWARDED_FOR, Cf-Connecting-Ip, X-Forwarded-For, User-Agent, Cookie	N/A
Fastly	Fastly-Client-IP, Fastly-Temp-XFF, X-Forwarded-For, User-Agent, Cookie	√ (Any Header)
Incapsula	HTTP_X_FORWARDED_FOR, X-Forwarded-For, Incap-Client-Ip, User-Agent, Cookie	N/A in Free Version
MaxCDN	X-Forwarded-For, User-Agent, Cookie	√ (Any Header)

C. CDN-proxied TCP port scanning

We examined CDNs' origin port configuration as shown in Table V [1]. We can see that some CDNs support configuring the port in a wide port range. Although some CDNs only support port 80 and 443, this port restriction can also be bypassed with the CDN-chaining trick.

TABLE V: *Origin's TCP Port Configuration*

	Port Configurable?	Port Range
Akamai	✓	Unspecified
Amazon	✓	1024-65535
Azure	✓	72,80-89,443-444...
Baidu	✗	80,443
Cloudflare	✗	80,443
Fastly	✓	Unspecified
Incapsula	✗	80,443
MaxCDN	✓	Unspecified

Therefore, an abuser can use CDNs to forward HTTP requests to any TCP port. What's more, we find that CDNs reply back different HTTP responses according to the origin port's status (filtered, closed or Open). As shown in Table VI, Fastly and Incapsula return different response according to the origin port's statuses. Though not all CDNs' responses are distinctive to differentiate the port's statuses, e.g., we can only differentiate "Open" from "Closed/Filtered" in Amazon CloudFront's responses, but by chaining CloudFront with Fastly, it is still applicable to differentiate these port statuses.

With such information disclosure, an abuser can employ the CDN as a TCP port scanning proxy. Further, the scan can be classified into two types:

- Host scanning: In the CDN's configuration, the abuser configures the target's IP or domain as the origin, and changes TCP port sequentially, to scan the target's different ports. However, this host scanning seems inapplicable, as the CDN's configuration change requires hours or even days to populate globally.
- Network Scanning: The abuser configures the origin to be a domain name under control and keeps the origin port unchanged. Then, the abuser can sequentially scan target IPs by resolving each DNS query to a different IP with a zero TTL in the DNS response.

Obviously, successful network scanning requires CDNs to respect the DNS TTL value, thus we measure these CDNs' DNS resolution behaviors. As shown in Table VII, in order to correctly associate each HTTP response with the IP we resolved in each DNS query, we could generally delay each scanning request over 60 seconds to wait stale DNS records invalidated.

[1]"Unspecified" means that we haven't found official document that illustrated the port range, however we have tested some ports to verify its configuration.

This CDN-based scan seems not different from normal proxy-based scan, but we think CDN IPs are less likely being blocked due to their high IP-reputation, and it is especially useful to scan a CDN-powered website which has a white-listed firewall to only allow the CDN's access. And it can be abused massively to speed up scanning process considering CDNs' global distribution and large quantity (See V).

D. TCP concurrent connection exhaustion attack

When a non-HTTP service is running on the port of website origin, such as TCP-based DNS, though the service can't parse HTTP protocol, abundant surrogates can still be abused to establish TCP connections with the service. These connections can surpass the service's TCP concurrent connection limit, and starve other real user's connection requests, resulting in a distributed connection exhaustion attack.

We test this attack against the DNS BIND server, as the BIND has a concurrent TCP connection limit which defaults to 100, and this limit is commonly shared between TCP-based DNS query and zone transfer [8]. Thus, surrogates can be abused to establish 100 concurrent TCP connections on port 53 of BIND server (issuing HTTP commands, not DNS commands), reaching the limit and preventing any new TCP connection to port 53 including DNS query and zone transfer.

We set up a DNS master server of BIND 9.8.2 and a DNS slave server of BIND 9.10.3, the slave server is set to synchronize periodically with the master server for DNS zone transfer. Both servers are firewall-protected by a strict TCP concurrent connection limit of 1 connection per IP only. When the master server is under attack, its "tcp-clients" limit of 100 is quickly overflowed, and errors are logged with the CDN's IP:

```
client x.x.x.x#30822: no more TCP clients: quota
    reached
```

When a normal client tries to query the DNS server with TCP at the same time, the connection can't be established. Meanwhile, zone transfer launched from the slave server fails too, logged as:

```
named[8334]: zone xxx.org/IN: Transfer started.
named[8334]: transfer of 'xxx.org/IN' from x.x.x.x
    #53: failed to connect: timed out
named[8334]: transfer of 'xxx.org/IN' from x.x.x.x
    #53: Transfer status: timed out
```

TABLE VI: *CDNs' HTTP Responses According To The Origin's TCP Port Status*

	Filtered (Null Response) H: HTTP header, B: HTTP Body	Closed (Rst Response) H: HTTP header, B: HTTP Body	Open (Complete 3-way handshakes) H: HTTP header, B: HTTP Body
Akamai	H: HTTP/1.1 504 Gateway Time-out B: An error occurred while processing your request.	H: HTTP/1.1 503 Service Unavailable B: Service Unavailable - Fail to connect	H: HTTP/1.1 503 Service Unavailable B: An error occurred while processing your request.
Amazon	H: HTTP/1.1 502 Bad Gateway B: CloudFront attempted to establish a connection with the origin, but either the attempt failed or the origin closed the connection	H: HTTP/1.1 502 Bad Gateway B: CloudFront attempted to establish a connection with the origin, but either the attempt failed or the origin closed the connection	H: HTTP/1.1 502 Bad Gateway B: CloudFront couldn't connect to the origin or the origin returned an incorrect response
Azure	H: HTTP/1.1 504 Gateway Timeout B: 504 - Gateway Timeout	H: HTTP/1.1 504 Gateway Timeout B: 504 - Gateway Timeout	H: HTTP/1.1 502 Bad Gateway B: 502 Bad Gateway
Baidu	H: HTTP/1.1 522 Origin Connection Time-out B: 522: Connection timed out	H: HTTP/1.1 521 Origin Down B: 521 - Origin Down	H: HTTP/1.1 520 Origin Error B: Web server is returning an unknown error
Cloudflare	H: HTTP/1.1 522 Origin Connection Time-out B: 522: Connection timed out	H: HTTP/1.1 521 Origin Down B: 521: Web server is down	H: HTTP/1.1 520 Origin Error B: 520: Web server is returning an unknown error
Fastly	H: HTTP/1.1 503 Connection timed out B: Error 503 Connection timed out	H: HTTP/1.1 503 Connection refused B: Error 503 Connection refused	H: HTTP/1.1 503 backend read error B: Error 503 backend read error
Incapsula	H: HTTP/1.1 503 Service Unavailable B: Error code 20 TCP connection timeout	H: HTTP/1.1 503 Service Unavailable B: Error code 8 TCP connection rejection (TCP Reset)	H: HTTP/1.1 503 Service Unavailable B: Error code 5 error in processing the server response
MaxCDN	H: HTTP/1.1 504 Gateway Time-out B: 504 Gateway Time-out	H: HTTP/1.1 502 Bad Gateway B: 502 Bad Gateway	H: HTTP/1.1 504 Gateway Time-out B: 504 Gateway Time-out

TABLE VII: *CDNs' DNS Resolution Behaviors*

	DNS Cache (Resolver)	Minimum TTL (second)
Akamai	per CDN node	≈60
Amazon	per data center	≈0
Azure	per CDN node	≈0
Baidu	per CDN node	≈60
Cloudflare	per data center	≈0
Fastly	per CDN node	≈0
Incapsula	N/A	N/A
MaxCDN	per CDN node	≈0

E. Circumvent Internet censorship

As CDNs have become an indispensable part of the Internet and served most of the popular websites (see Figure 4), it's impossible for Internet censorship systems to block CDNs without causing collateral damage [9], [10].

Therefore, on the fact that surrogates are not blocked and located outside the censorship system, CDNs can be used to access the censored websites. Such circumvention can be conducted in just 3 steps:

1) Register a new domain name (e.g., ".tk" and ".ml"), to ensure this domain name can escape DNS spoofing [11].
2) Apply for CDN service with the registered domain name, and configure the censored website as the origin.
3) Access the registered domain with HTTPs[2], this adds another layer of security and helps to bypass on-path DPI (Deep Packet Inspection) [12] between users and the CDN.

However, in the CDN-retrieved web pages, there may exist absolute URLs containing the censored domain names. But this problem can be solved with a pre-installed browser extension to substitute the absolute URLs with the newly registered domain name, then following clicks on these URLs can hit the CDN service again.

Holowczak *et al.* [4] discussed CDN-based circumvention too, but it needs a side channel server to bypass DNS spoofing, which could be a single point of failure once being blocked. And [4] can only access websites hosted on CDNs already, there is still 25.75% of the Alexa Top 1000 websites haven't been hosted on CDN yet as shown in Section V Figure 4.

[2]CDNs offer user-facing HTTPs connections even if back-end websites only support clear-text HTTP.

As not all the censored websites have been hosted on CDNs already, we believe this paper's trick is more flexible for end users. An end user can register a CDN service to circumvent different censorship systems accordingly, and choose other CDNs when one CDN is blocked [13].

F. CDN-based DoS attack against any website

On the fact that the CDN's user-facing connection and website-facing connection can have inconsistent TCP status and asymmetric bandwidth, Triukose *et al.* [3] presents a bandwidth DoS attack against CDN-powered website servers. In the attack, an attacker sends a large number of requests to different surrogates directly, requesting for the same large file on the website. The URL of these requests is appended with a random query string to bypass the CDN cache. Then, the attacker cuts off all front-end connections, while the CDN still sustains all back-end connections that transfer in a higher bandwidth. This results in a bandwidth exhaustion DoS attack against the website.

TABLE VIII: Whether CDNs Keep Back-end Connection While A Attacker Manipulates Front-end Connection

	Front-end Connection Close	Front-end Connection Slow Read
Akamai		Keep
Amazon	Close	Keep
Azure	Close	Keep
Baidu	Close	Keep
Cloudflare	Close	Keep
Fastly		Keep
Incapsula	Close	Keep
MaxCDN	Close	Keep

However, in experiments we find that some CDNs have ignored the query string in the URL of requests by default, thus requests of the same file with random query string URLs won't be forwarded back to the website server again. And some CDNs will close back-end connections accordingly once front-end connections are cut off, as shown in Table VIII. These mitigations can all invalidate the attack in [3], besides, this attack is only applicable to websites deployed behind CDNs already.

Differently, with the CDN origin abuse, an attacker can point the origin to any websites, and customize CDNs' cache policy directly, e.g., whether to forward requests based on the URL's path, whether to filter the URL's query string. Further, with *Sockstress* [6] trick to slowly read responses from surrogates[3], this can still result in a back-end bandwidth exhaustion attack, while the attacker's front-end bandwidth consumption is kept at the minimum.

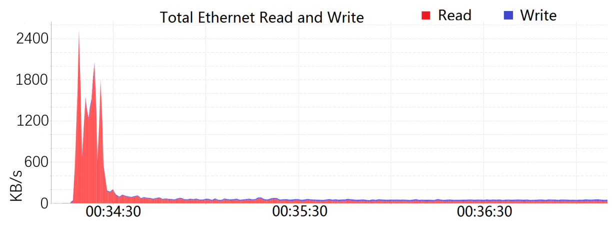
In order to evaluate the attack, we deploy a virtual web server in our lab with 100 Mbps bandwidth. The website is setup with Nginx 1.10.0's default configuration, and protected with IPtable by limiting the incoming TCP connections per IP to be 30.

Then we register CloudFront's service by configuring our website as the origin. In order to minimize any potential damage to the CDN, we only use 50 surrogates. From a PC as the attacker, we directly send 30 requests to each chosen surrogate, all requesting for a 10MB file in our website, with random query string URL to bypass the CDN cache.

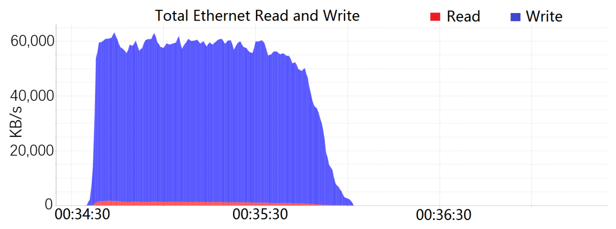
Fig 3(a) and Fig 3(b) compare the bandwidth consumption. After connecting the surrogates, attacker's front-end bandwidth is manipulated within 100 Kbps with *Sockstress* trick, while website's bandwidth achieves 60 Mbps at maximum.

Fig. 3: DoS Attack of Asymmetric Bandwidth Exhaustion

(a) Attacker Bandwidth



(b) Website Bandwidth



We can see that the more surrogates are abused, this attack can lead to a more severe bandwidth exhaustion. Considering CDN's large computational resources and high bandwidth, we believe this bandwidth exhaustion attack could be a disaster for websites, especially for small websites most of which haven't been hosted on CDN yet.

G. Internet survey fraud through CDNs

We create an Internet survey demo at "polldaddy.com", and allow vote from one IP only once. We managed to bypass this one-vote-per-IP limit by voting through different surrogates, and we succeeded in voting hundreds of times.

[3]The attacker can set the front-end connection's TCP window size to be zero or a small value.

Note that the CDN's user-facing IP (inbound IP) is different from the website-facing IP (outbound IP). With experiments in Section V, we found that CDNs will group received requests from different inbound IPs, and use fewer outbound IPs to pull the requested resources back. Thus, in order to bypass such outbound IP related restriction, an abuser can just randomly schedule as many inbound IPs as possible for simplicity, or record the mapping between inbound IPs and outbound IPs with long-term measurement.

We agree that survey systems can use authentication to solve this IP-based survey fraud easily, but it's still useful in cases when massive IPs are needed first.

H. Bypass web service's IP Geo-blocking

Due to the copyright, websites providing online video or music service normally restrict their service to a certain country or area only, and this restriction is simply enforced based on the end-user's IP Geo-location.

As CDN surrogates are globally distributed, this IP Geo-blocking can be bypassed directly. We experiment with an online radio service provider *Pandora Media, Inc*. Though *Pandora* provides its music service only to end-users in the U.S., Australia and New Zealand, we configure "www.pandora.com" as the CDN origin and successfully enjoy *Pandora* service outside the permitted Geo-location.

In Section V, by finding millions of surrogates and verifying their global distribution, we believe this CDN-based Geo-restriction bypassing is more beneficial than public proxy-based bypassing, especially considering CDN's high bandwidth and high IP reputation.

V. CDNs' Characteristic Distributions

From previous sections, we can see that in order to schedule which and how many surrogates to be abused, an abuser has to gather surrogates' IP addresses. And the number and distribution of surrogates determines the severity of aforementioned abuses. However, to the best of our knowledge, this information of surrogates is not fully publicized on the Internet, but we find that it can be revealed due to CDNs' unintentional information exposure through HTTP headers or sub-domain names. Based on such exposure, we perform a real-world measurement to show the CDNs' IP distribution, and we also examine the CDN utilization for Alexa Top 1000 websites to demonstrate that CDNs are widely used.

We describe CDN's user-facing IP as inbound IP, and website-facing IP as outbound IP. In order to find inbound IPs, an Internet-wide scan is a straightforward method. By setting HTTP requests' "Host" header to be a pre-known CDN customer's domain name, an Internet-wide HTTP scanning can be used to find IPs which return correct web pages. However, this method cannot filter out open proxies, and scanning may be deemed offensive. Inbound IPs can also be gathered by querying open DNS resolvers for a CDN customer's domain name [14], [15], [16]. As a CDN returns the inbound IP that's close to the LDNS (local DNS) in network proximity, and millions of globally distributed open recursive DNS resolvers can be used to query [17]. Although this may raise ethical concerns by directly using the open resolves found by active Internet-wide measuring.

Thus, different from actively scanning or querying, we propose following two passive methods to find inbound IPs:

- Analyzing the "Censys" [18] data: Website "censys.io" periodically launches Internet-wide HTTP scanning by setting the "Host" header to be the scanned IP. We find that different CDNs return distinguished HTTP responses when receiving such IP-based "Host" header, as shown in Table IX. Thus inbound IPs can be filtered directly from "Censys" data.

TABLE IX: Characteristic of HTTP Response

CDN	HTTP Status Code	HTTP Response (H: Header B: Body)
Akamai	400	H: "Server: AkamaiGHost"
Amazon	403	H: "Server: CloudFront"
Azure	404	H: "Server: ECCAcc"
Baidu	403	H: "Server: yunjiasu-nginx"
Cloudflare	403	H: "Server: cloudflare-nginx"
Fastly	500	H: "X-Fastly-Request-ID: "
Incapsula	503	H: "X-Iinfo:", B: "Incapsula incident ID"
MaxCDN	430	H: "Server: NetDNA-cache/2.2"

- Analyzing the passive DNS data: CDNs use CNAME to direct their customer's domain to the CDN-assigned sub-domains as shown in Table X, these domain suffixes can be used to find inbound IPs from the passive DNS data.

TABLE X: CDN Assigned Sub-Domain

CDN	Assigned CNAME	CDN	Assigned CNAME
Akamai	.akamai.net .akamaiedge.net .akadns.net .edgesuite.net	Cloudflare	N/A (CNAME unsupported)
Amazon	.cloudfront.net	Fastly	.global.prod.fastly.net
Azure	.azureedge.net	Incapsula	.incapdns.net
Baidu	N/A (CNAME unsupported)	MaxCDN	.thu.netdna-cdn.com

Although these techniques may only give a lower bound of inbound IPs, we still find millions of inbound IPs as shown in Table XI, and it's already abundant for the abuses in this paper. In Table XI, we find that the number of accessible inbound IPs varies each time we scan. We speculate that these varying IPs may be the result of the CDN's on-demand scheduling policy, or some IPs are reserved to offer service to higher-rank customers who have paid for CDN service.

Besides that, we also find CDNs will group incoming requests and forward these requests to the customer website with a small list of outbound IPs. Therefore, in order to find more outbound IPs, we sequentially deploy 3 websites as the origin (located in Beijing, California and Singapore). For each origin, we use the "Censys" data to filter possible inbound IPs, and we use ZMap [19] to verify the accessibility of these inbound IPs. Then, for each accessible inbound IP, we send an HTTP request with random URL to bypass the CDN's cache mechanism. Meanwhile, we capture the outbound IPs that connect to our website at each origin.

In Table XI, we captured a different set of outbound IPs for each origin. After merging the 3 sets of outbound IPs without duplicates, we get the total outbound IPs. It can be seen that the number of outbound IPs is much fewer compared with inbound IPs, we infer that CDN's optimization strategy is mainly focused on the user-facing front-end connection. Inbound IPs' mass distribution can help to minimize front-end's delay, and the cache mechanism help to reduce the back-end's delay. Thus, CDNs can deploy a few outbound nodes scattered globally which have optimized bandwidth, and this small outbound IP list can be easily white-listed in the customer websites' firewall as an origin-shielding defense. However, this defense still can't protect these customer websites, if surrogates themselves are abused to proxy the attack.

TABLE XI: CDNs' IP Distribution

	Origin Location	Inbound IPs	Outbound IPs	Total Outbound IPs
Akamai	Beijing	2976272	51412	78725
	California	2210349	40177	
	Singapore	2255356	31148	
Amazon	Beijing	139822	475	2368
	California	129033	256	
	Singapore	101541	1859	
Azure	Beijing	2052	597	645
	California	2057	55	
	Singapore	2049	53	
Baidu	Beijing	8719	39	104
	California	9069	34	
	Singapore	9143	36	
Cloudflare	Beijing	274014	77	185
	California	263338	53	
	Singapore	223023	55	
Fastly	Beijing	50676	693	793
	California	49115	773	
	Singapore	40919	772	
Incapsula	Beijing	46838	130	232
	California	53328	178	
	Singapore	39377	177	
MaxCDN	Beijing	6190	12	15
	California	5540	13	
	Singapore	6202	12	

The number of outbound IPs affects the abuse cases that count on this number, e.g., Internet survey fraud. Thus, an abuser may have to measure and record the mapping between inbound IPs and outbound IPs when launching such abuse on a massive scale. We leave the details of this mapping for future evaluation.

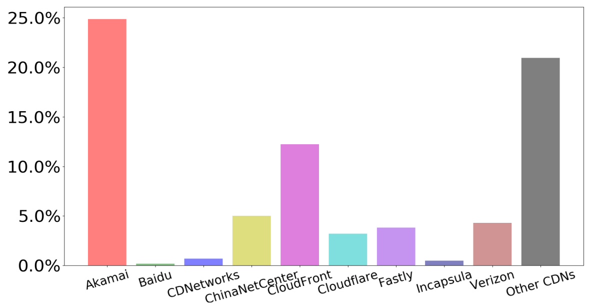


Fig. 4: CDN utilization for Alexa Top 1000 websites

To demonstrate CDNs' widely utilization on the Internet, we have also identified the CDN usage ratio for Alexa top

1000 websites, as shown in Figure 4. Our results confirm Akamai's dominant role in the CDN market [1], and show that more than 74% of the Alexa top 1000 websites use the CDN service. We can see that the CDN has played a critical and central role in the Internet infrastructure. Once CDNs are abused, it is harder to defend with IP blocking, which can result in collateral damage for these popular websites.

VI. Discussion

A. Anonymity and Cost

As illustrated in the paper, attackers are able to abuse CDNs in various ways. However, it will discourage attackers when the CDN needs to be paid or could expose attackers' identity.

Unfortunately, as in Table XII, CDNs provide the convenient "free" or "free-trial" services to their potential customers, and thus to attackers as well. For registration requirements, five out of eight CDNs only require a valid email address to confirm the registration. Akamai (Exceda) and Amazon require valid credit card details, attackers may still keep anonymous with stolen credit cards. Azure (China) requires a valid phone number to receive SMS, it can be anonymous too. Note that Akamai doesn't accept registration from individual customers, we overcome this limitation through public third-party re-seller "xcdn.exceda.com".

TABLE XII: CDN Registration Requirements and Cost

CDN	Register Requirement	Price	Anonymity
Akamai	Email address Credit card	Free trial	✓
Amazon	Email address Credit card	Free trial	✓
Azure (China)	Email address Phone number	Free trial 1CNY	✓
Baidu	Email address	Free Service	✓
Cloudflare	Email address	Free Service	✓
Fastly	Email address	Free Service	✓
Incapsula	Email address	Free Service	✓
MaxCDN	Email address	Free trial	✓

B. Responsible Disclosure

It is obvious that our research brings up several ethical concerns. However, we controlled experiments to avoid impact on the operation of CDNs by throttling consumed CDN bandwidth and attacking our own website. Also, we attempted to contact all 8 CDNs by sending our findings through the email address found on their websites. 7 of 8 CDNs replied, and the responses are summarized as bellows.

Amazon CloudFront: AWS security team acknowledged us for our responsible report. They had an internal discussion with the CloudFront security team, and they further contacted us for more techniques details.

MaxCDN: MaxCDN believed that they were not adequately addressing the issues we reported. And after reviewing this further, they were aware of this issue and were working through how to best resolve it.

Fastly: Their security team acknowledged us for our disclosure, and contacted us to share more experimental details. They have begun to investigate these issues internally.

Cloudflare: Cloudflare acknowledged our findings and had a discussion with us. They stated that most of the attacks were interesting. They also informed us that, the fact that Cloudflare doesn't respect DNS TTL was beyond their expectations, and they were considering mitigations.

Baidu: Baidu security team contacted us about the technical details. Also, they confirmed that the issues in origin validation.

Akamai: Akamai's security researcher acknowledged many CDNs have limitations with respect to verifying the ownership of customer assets. But they do not believe this issue stem from a fundamental flaw with Akamai's architecture, or with CDNs in general. Nevertheless, Akamai is well-aware of any potential risks related to insufficient ownership verification, and they prefer to utilize numerous performance analytics techniques to continuously monitor customer origin health.

Incapsula: They acknowledged us for disclosure, but they didn't provide any further comment to date.

We receive no further comments about the solving progress until now.

C. Mitigation

The easiest defense is to increase the economic cost of CDN abuse, e.g., stop the "free" service, although this may inapplicable due to CDNs' market competition. Thus we discuss only technical mitigations.

1) Validate the Origin and Enforce the Origin Pinning: The root cause of CDN abuses is that CDNs offer rich configurable options but fail in validating the customer-supplied information, especially the ownership of origin.

We recommend CDNs to enforce the validation of domain and IP as a pair to confirm the ownership of the origin. E.g., CDNs can require their customers to both configure the origin's domain and IP in the web user interface, and upload a special file to the origin website. Besides that, CDNs should continuously monitor the change of this domain-IP pair, or to constantly probe this special file's existence. Any abnormality will make CDNs terminate the service and relaunch a new validation process, we call this process the *Origin Pinning*.

2) Strict Inspection of the Configuration Options: As CDNs offer configuration options but lack strict inspection, a CDN customer or an adversary can configure these options to meet various abuse needs. We believe there still exist other possible abuses not listed in this paper. It is preferable for CDNs to enforce strict inspection upon customer-supplied options, especially for options that may pose security threats against the origin, *e.g.*, white-listing the origin ports, forbidding modification of the "Host" header, or selectively supporting HTTP headers filtering. In a word, CDNs should not sacrifice security for flexibility and functionality.

VII. Related Work

CDN security is an emerging area. Previous researches have largely focused on identifying security vulnerabilities or issues introduced by CDNs, including the growing complexity of the

CDN ecosystem [20] and the integrity of web content through untrusted CDNs [21], [22].

The evolving deployment of CDN also brings challenges to the contemporary Internet ecosystem, especially for HTTPS certificate deployment. The end-to-end nature of HTTPS and the man-in-the-middle nature of CDN result in a fundamental semantic conflict, and this leads to various problems with current HTTPS practice adopted by CDNs, such as private key sharing and insecure back-end communication [23], [24]. Further, the good reputation and indispensability of CDNs could be abused to circumvent Internet censorship, and "Domain fronting" and "CacheBrowser" were proposed to race with the evolving censorship systems [4], [25], [26].

The vulnerabilities that could be abused to conduct DoS attack are also found to exist in CDNs, which not only cause damage to CDNs' customer websites [3] but also to CDNs themselves [27]. Comparing with this study, we extend the work in [3] to attack *any* websites that haven't been deployed behind CDNs yet. CDN forwarding-loop attack, chaining different CDNs into a loop, causing the request to be processed repeatedly and resulting in a DoS attack against CDNs also found in previous research [27]. Several "origin-exposing" attack vectors that can be used to discover the IP address of the website origin are discussed in [28], [29], and CDN's surrogate mappings can be maneuvered with crafted DNS records in [30].

In this study, we revisit the topic of CDN abusing. Our study is much more comprehensive in terms of abusing categories, and it serves as a complement to existing studies.

VIII. Conclusion

The CDN is a critical Internet infrastructure now. To minimize deploy efforts, CDNs provide their customers with rich configuration options but lack sufficient validation, especially for the origin option. Through a comprehensive practical analysis, our study shows attackers could abuse the lack of origin validation to conduct various CDN abuses. We also discuss mitigations to defense such abuses. However, the coordination among CDNs is still needed. We envision our work can motivate CDNs to fully evaluate their customer-supplied configuration options to remove security flaws.

Moving forward, we believe that CDNs' implementation or operational weaknesses can indeed pose new threats to the Internet security, and there may still exist other attacks of leveraging CDNs' inconsistencies in between deploying policies. Further research is needed to get a more comprehensive view of the services and develop effective solutions to mitigate CDNs' security threats.

IX. ACKNOWLEDGMENTS

This research is sponsored by the National Natural Science Foundation of China (grant 61472215, U1636204).

References

[1] AkamaiTech. (2017) Facts & figures. [Online]. Available: https://www.akamai.com/us/en/about/facts-figures.jsp

[2] S. Systems, O. Spatscheck, A. Barbir, and R. Nair, "Known Content Network (CN) Request-Routing Mechanisms," RFC 3568, Oct. 2015.

[3] S. Triukose, Z. Al-qudah, and M. Rabinovich, "Content Delivery Networks : Protection or Threat," *ESORICS '09*, 2009.

[4] J. Holowczak and A. Houmansadr, "CacheBrowser : Bypassing Chinese Censorship without Proxies Using Cached Content," *CCS '15*, 2015.

[5] J. Lemon, "Resisting syn flood dos attacks with a syn cache," *USENIX BSDCon '02*, 2002.

[6] S. Sergey, "Tweaking to get away from SlowDOS," *OWASP*, 2012.

[7] A. Petersson and M. Nilsson, "Forwarded HTTP Extension," RFC 7239, Jun. 2014. [Online]. Available: https://rfc-editor.org/rfc/rfc7239.txt

[8] BIND. ([Accessed Nov. 2016]). [Online]. Available: http://www.bind9.net/arm98.pdf

[9] N. Jones, M. Arye, J. Cesareo, and M. J. Freedman, "Hiding amongst the clouds: A proposal for cloud-based onion routing," *FOCI*, 2011.

[10] C. Brubaker, A. Houmansadr, and V. Shmatikov, "Cloudtransport: Using cloud storage for censorship-resistant networking," *PETS '14*, 2014.

[11] Anonymous, "Towards a comprehensive picture of the great firewall's dns censorship," *FOCI '14*, 2014.

[12] Z. Wang, Y. Cao, Z. Qian, and S. V. Krishnamurthy, "Your State is Not Mine: A Closer Look at Evading Stateful Internet Censorship," *IMC '17*, 2017.

[13] R. Ensafi, D. Fifield, P. Winter, N. Feamster, N. Weaver, and V. Paxson, "Examining How the Great Firewall Discovers Hidden Circumvention Servers," *IMC '15*, 2015.

[14] C. Huang, W. Angela, J. Li, and K. Ross, "Measuring and Evaluating Large-Scale CDNs," *IMC '08*, 2008.

[15] F. Streibelt, J. Böttger, N. Chatzis, G. Smaragdakis, and A. Feldmann, "Exploring edns-client-subnet adopters in your free time," *IMC '13*, 2013.

[16] F. Chen, R. K. Sitaraman, and M. Torres, "End-User Mapping: Next Generation Request Routing for Content Delivery," *ACM SIGCOMM '15*, 2015.

[17] M. Kührer, T. Hupperich, J. Bushart, C. Rossow, and T. Holz, "Going Wild: Large-Scale Classification of Open DNS Resolvers," *IMC '15*, 2015.

[18] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman, "A Search Engine Backed by Internet-Wide Scanning," *CCS '15*, 2015.

[19] Z. Durumeric, E. Wustrow, and J. A. Halderman, "ZMap: Fast Internet-wide Scanning and Its Security Applications," *USENIX Security '13*, 2013.

[20] S. Volker, S. Georgios, L. William, and B. Steven, "The growing complexity of content delivery networks: Challenges and implications for the internet ecosystem," *Telecommunications Policy Journal*, 2017.

[21] A. Levy, H. Corrigan, and D. Boneh, "Stickler : Defending against Malicious Content Distribution Networks in an Unmodified Browser," *IEEE S&P*, 2016.

[22] Y. Gilad, A. Herzberg, M. Sudkovitch, and M. Goberman, "CDN-on-Demand : An Affordable DDoS Defense via Untrusted Clouds," *NDSS '16*, 2016.

[23] J. Liang, J. Jiang, H. Duan, K. Li, T. Wan, and J. Wu, "When HTTPS meets CDN: A case of authentication in delegated service," *IEEE S&P*, 2014.

[24] F. Cangialosi, T. Chung, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson, "Measurement and Analysis of Private Key Sharing in the HTTPS Ecosystem," in *CCS'16*, 2016.

[25] H. Zolfaghari and A. Houmansadr, "Practical Censorship Evasion Leveraging Content Delivery Networks," *CCS '16*, 2016.

[26] D. Fifield, C. Lan, R. Hynes, P. Wegmann, and V. Paxson, "Blocking-resistant communication through domain fronting," *PETS '15*, 2015.

[27] J. Chen, J. Jiang, X. Zheng, and H. Duan, "Forwarding-Loop Attacks in Content Delivery Networks," *NDSS '16*, 2016.

[28] T. Vissers, T. V. Goethem, W. Joosen, and N. Nikiforakis, "Maneuvering Around Clouds : Bypassing Cloud-based Security Providers," *CCS '15*, 2015.

[29] L. Jin, S. Hao, H. Wang, and C. Cotton, "Your Remnant Tells Secret : Residual Resolution in DDoS Protection Services," *DSN '18*, 2018.

[30] S. Hao, C. Uc, S. Diego, and H. Wang, "End Users Get Maneuvered : Empirical Analysis of Redirection Hijacking in Content Delivery Networks," *USENIX Security '18*, 2018.