CacheBrowser: Bypassing Chinese Censorship without Proxies Using Cached Content

John Holowczak and Amir Houmansadr
College of Information and Computer Sciences
University of Massachusetts Amherst
jholowcz@umass.edu amir@cs.umass.edu

ABSTRACT

The cached Internet content served by content delivery networks (CDN) comprises a large fraction of today's Internet traffic, yet, there is little study on how real-world censors deal with blocking forbidden CDN-hosted Internet content. We investigate the techniques used by the Great Firewall of China to block CDN-hosted content, and demonstrate that blocking CDN content poses unique technical and non-technical challenges to the censors. We therefore design a client-side circumvention system, *CacheBrowser*, that leverages the censors' difficulties in blocking CDN content. We implement CacheBrowser and use it to unblock CDN-hosted content in China with a download latency significantly smaller than traditional proxy-based circumvention systems like Tor. CacheBrowser's superior quality-of-service is thanks to its *publisher-centric* approach, which retrieves blocked content directly from content publishers with no use of third-party proxies.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and protection*; E.3 [**Data**]: Data Encryption

General Terms

Algorithms, Design, Security

Keywords

Censorship resistance; Unobservability; CDN; Content cache

1. INTRODUCTION

Internet censorship continues to remain a global threat to the freedom of speech and open access to information. This is largely due to the Internet's old "end-to-end communication" paradigm borrowed from the telephony system of the 70's: censors can prevent users *at low cost* from making end-to-end connections with forbidden content publishers identified by unique IP addresses. This is enforced either by blocking the IP addresses of forbidden publishers or by preventing users from learning such IP addresses, i.e., through DNS interference [30].

Fortunately, the Internet has been gradually diverging from the old end-to-end communication paradigm by embracing modern forms of communication. In particular, *caching Internet content*, practiced to improve performance, security, and reliability, has revolutionized the way content is being communicated on the Internet. Content caching is particularly being deployed pervasively by content delivery networks (CDN) to fulfill end-users' demands for high-bandwidth, low-latency access to popular content like video streams and online applications. The Akamai CDN provider alone serves more than 30% of the Internet's static web traffic [34], accompanied by other competitor CDN providers [8] offering a broad range of service plans and performance guarantees. The success story of content caching in CDN networks has even inspired several next-generation Internet architectures [2, 33, 36] with content caching as their main design principle, and content caching has become an inseparable part of file sharing overlay networks like BitTorrent.

In this paper, we take the first steps in studying the impacts of *content caching* on the Internet censorship problem. We analyze the effectiveness of existing censorship techniques in blocking content cached on content delivery networks (CDN), and investigate how the Great Firewall of China (GFW), unanimously believed to be the most competent censorship architecture, blocks forbidden CDN content in-the-wild. Our study finds that *content caching by CDN networks poses significant technical and non-technical challenges to the censors*, even despite self-censorship by a major CDN provider, Akamai, as discovered in our study. Specifically, we show that IP address blocking, a highly-effective technique in blocking regular Internet content, is impractical in blocking CDN content, and that the GFW refrains from IP blocking CDN content due to the potential collateral damage. This is because a typical CDN content object is hosted on many (e.g., several thousands of) IP addresses that are shared among many (potentially hundreds of thousands) non-related content publishers, therefore, any IP address blocking attempt to censor the forbidden CDN content will equally censor the non-related —potentially non-forbidden —content publishers.

CCS'15, October 12–16, 2015, Denver, Colorado, USA.

ACM 978-1-4503-3832-5/15/10 ...$15.00.
DOI: http://dx.doi.org/10.1145/2810103.2813696.

Based on the lessons of our analysis, we design a circumvention system, *CacheBrowser*, that exploits the censors' difficulties in censoring CDN content. CacheBrowser evades censors' DNS interference by directly retrieving blocked CDN content from one of the many (e.g., tens of thousands) CDN servers hosting the same blocked CDN content. We have built CacheBrowser as a client-side software, and used it to circumvent the GFW by accessing popular forbidden CDN-hosted websites like `change.org` and `facebook.com`.

Contrary to proxy-based circumvention systems like Tor [18] and Psiphon [27], CacheBrowser takes a new circumvention approach, which we call *publisher-centric*. In this approach, end-users make no use of third-party proxies to retrieve censored Internet content, but instead obtain the censored content directly from the content publishers. We show that CacheBrowser's publisher-centric approach has significant advantages over traditional proxy-based circumvention systems, e.g., CacheBrowser's download latency is drastically smaller than Tor.

Due to its publisher-centric design, CacheBrowser can only unblock CDN-hosted content (because non-CDN content can be IP address filtered). Fortunately, this is not a real-world limitation since we show that a significant fraction of popular websites blocked in China are already hosted on CDN. Also, hosting content on CDN is seamless and affordable even for personal websites and blogs [10,12,13], therefore, a censored content publisher can unblock her content by transitioning to a CDN platform.

In short, we make the following main contributions:

- We make the first comprehensive analysis of how content cached by CDN networks can be censored, and how the GFW is blocking such cached content.
- We design a system called CacheBrowser that leverages the censors' challenges in blocking CDN content.
- We implement CacheBrowser and demonstrate how it can be used to unblock popular censored CDN content in China, e.g., `facebook.com`.
- We introduce the publisher-centric approach for circumvention, which has significant advantages over traditional proxy-based systems like Tor.

2. BACKGROUND: CONTENT DELIVERY NETWORKS

A content delivery network (CDN) is a *distributed system* run by a **CDN provider** that delivers Internet content, i.e., **CDN content**, from content publishers, referred to as **CDN customers**, to **end-users**. A typical CDN network caches content of its customers on multiple (usually many) servers located at disperse geographic locations, and directs an end-user looking for the cached content to a caching server that optimizes user experience, e.g., reduces download latency. In addition to the improved QoS for end-users, CDN hosting makes a content publisher's service more scalable and reliable, saves the publisher money by reducing the load on their network backbone, and boosts pageranking on search engines [11]. Therefore, an increasing number of content publishers, ranging from small personal webpages and blogs [10] to large-scale social networking websites, are hosting their content on CDN systems.

The CDN market is shared between an increasing number of competitor commercial CDN providers [8]. Akamai has the largest market share by serving [1,34] nearly 30% of all web traffic, communicated over several millions of HTTP requests per second. CDNs have become such a key part of the Internet that even their short-term failure severely impacts the Internet as a whole.[1]

2.1 CDN Architecture

Despite the variety of CDN providers [8], they share the same high-level architecture [35].

Edge servers: An edge server is a computer server run by a CDN provider to serve the CDN provider's customer content to end-users. Commercial CDN networks are composed of very large numbers of edge servers located at diverse geographic locations to ensure efficient distribution of cached content. For instance, Akamai runs 170,000 edge servers scattered across 102 countries [1]. Upon a user request for cached content, the queried edge server will either respond the content from its local cache, or request the content from the content publisher.

Mapping system: Every CDN network has a mapping system that directs querying users to the "best" edge server hosting the requested content. Mapping the best edge servers to the querying end-users is a complex, dynamic process and takes into account various factors including the geographic location of the end-user, the global network conditions, the load on edge servers, etc. Ideally, the mapping system will return an edge server that maximizes the QoS for each querying user, e.g., one with a close proximity to the querying user.

Other components: A typical CDN network is composed of other key components [35], that we exclude due to their little relevance to our study. In particular, each CDN network has a *transport system* that transmits customer content to edge servers, a *communication and control system* that exchanges status information, configuration updates, and control messages within the CDN system, and a *management portal* used by CDN customers to control how their content and applications are being served by the hosting CDN system.

2.2 Typical CDN Communication

Suppose that an end-user, `CDNUser`, decides to access the HTTP content at URL_R=`http://www.cnn.com/index.html`, which belongs to `CDNCustomer` (the CNN in our case). Also, suppose that `CDNCustomer` is a CDN customer of `CDNProvider` (Fastly CDN for the given example). That is, `CDNCustomer`'s content is served by `CDNProvider`. The following steps are taken in order for `CDNUser` to obtain the requested HTTP content (URL_R) hosted by `CDNProvider`:

1. Generic DNS lookup: `CDNUser` will send a DNS lookup request to a generic top-level domain (TLD) DNS server for URL_R's domain name. The generic TLD resolver will be redirected to one of the `CDNProvider`'s top-level name server (TLNS) as the DNS authority, e.g., a Fastly TLNS server. Such redirection is enforced by listing the TLNS server of `CDNProvider` in the "authorities" section of the DNS record for the queried domain name, often with a large TTL value.

2. CDN DNS lookup: The CDN TLNS returned in the previous stage will delegate the DNS query to some low-level Name Servers (LLNS) of `CDNProvider` with a shorter DNS TTL. The DNS query is finally sent to one of these

[1] `https://gigaom.com/2011/08/08/akamai-dns-issue/`

CDN LLNS servers, which will return the IP address of a `CDNProvider` edge server that will serve the content for URL_R. The edge server is chosen by a complex mapping algorithm [34,35] taking into account factors like geographic proximity, availability of content, network bandwidth, etc. Edge server assignments are extremely dynamic (to optimize the performance of content delivery), i.e., the edge server returned to a `CDNUser` querying URL_R may *change roughly every minute* [34].

3. Requesting CDN edge server for content: Finally, `CDNUser` will send an HTTP/HTTPS GET request to the returned `CDNProvider` edge server for URL_R. If the content is not already cached by the edge server, the edge server will first retrieve it from `CDNCustomer`'s *origin servers* [34].

2.3 Types of CDN Networks

In a **shared CDN (S-CDN)** network the CDN infrastructure (e.g., edge servers, mapping system, and communication and control system) is shared among several (usually many) CDN customers, i.e., content publishers. This type of CDN is mainly operated by commercial CDN providers like Akamai, CloudFlare, and Fastly. Content publishers (e.g., the CNN or Yahoo!) pay commercial S-CDN providers for hosting their content or applications. On the other hand, a **private CDN (P-CDN)** infrastructure is dedicated to serve content delivery for a specific content publisher. The use of a private CDN can provide better control on content distribution and better privacy protection, however, its maintenance (e.g., ensuring security, reliability, and availability) may be complicated, costly, and prone to failures. Therefore, except for giant content producers, typical content publishers tend to use shared CDN services powered by commercial CDN providers. Also, some private CDN networks are powered by commercial CDN providers.

3. CENSORSHIP OF CDN CONTENT

Despite CDN traffic comprising a significant fraction of Internet communications, there is no study on how real-world censors block forbidden CDN content. We study how content caching by CDN networks impacts Internet censorship. We start by evaluating traditional censorship techniques against CDN content, then, study how China blocks CDN content through in-the-wild experiments. The challenge to our study is the lack of knowledge on the internal specifications of in-the-wild censors, real-world enterprise CDN systems, and possible business agreements between the censors and CDN providers.

3.1 Possible CDN Censorship Techniques

Considering the architecture of typical CDN networks, introduced in Section 2, we discuss how various censorship techniques can be leveraged to block CDN content.

3.1.1 IP Address Filtering

IP address filtering is a simple, low-cost technique for censorship. In this approach, censors identify (then, blacklist) the IP addresses of forbidden content publishers (e.g., through browsing their websites), and drop network packets to such blacklisted IP addresses through null routing on gateway routers of the censoring ISPs [3].

IP address blocking is one of the most widely used censorship techniques, and most commercial network devices (e.g., routers and firewalls) have the built-in capability for performing it with low computation overheads. However, we argue that **IP address filtering is highly incapable of blocking CDN content since it causes significant collateral damage to the censors.** This is due to the following properties of CDN traffic:

P1: Pervasive, distributed caching– In order to ensure availability and optimize performance, a CDN system caches customers' content on large numbers of edge servers, residing at various, usually dispersed, Internet locations. To IP-filter CDN content, censors will need to enumerate *all* the IP addresses caching the forbidden content, and then blacklist all of them. This is contrary to regular (non-CDN) Internet content in which case blocking a single (or a few) IP addresses suffices to entirely block a forbidden publisher.

P2: Shared IP addresses– Commercial CDN providers share their infrastructure, e.g., edge servers and mapping system, between all (potentially many) of their CDN customers. Therefore, a forbidden CDN content will be retrieved from the same set of (edge server) IP addresses as other non-forbidden CDN content served by the hosting CDN provider. As a result, any attempt to IP filter forbidden CDN content will equally affect the non-forbidden CDN content hosted by the hosting CDN provider. Note that this does not apply to P-CDN systems.

P3: Highly dynamic IP assignment– To optimize load balancing and quality-of-service, mapping edge servers to end-users is performed highly dynamically, e.g., Akamai's mapping system updates returned IP addresses as frequently as every minute [34]. Therefore, censors will not be able to map specific forbidden CDN content to a small subset of edge server IP addresses, even within short time intervals.

3.1.2 DNS Interference

Along with IP address filtering, DNS interference is one of the most widely practiced censorship techniques [3,4,30]. In this approach, censors prevent end-users from learning the IP addresses that serve forbidden content, i.e., they interfere with the name resolution process. This may be implemented in different ways: dropping DNS messages that query forbidden domain names, hijacking DNS communications, and poisoning DNS records on DNS servers [3,4].

DNS interference is highly effective [4] in censoring Internet content as name resolution is an essential part of Internet communications, both for CDN and non-CDN content. However, DNS interference can be easily circumvented if a blocked user bypasses the name resolution process and learns the IP addresses serving forbidden domain names from other means, e.g., out-of-band channels. This is why real-world censors usually complement DNS interference with IP filtering. However, as discussed above, IP filtering is ineffective (i.e., causes collateral damage) in blocking CDN content.

3.1.3 Keyword/URL filtering using DPI

Modern network monitoring equipment can inspect network traffic searching for forbidden keywords or URLs, a capability known as deep packet inspection (DPI) [30]. A network flow found to contain a forbidden keyword or URL is then interfered with by the censors, e.g., by dropping packets or forging TCP RST messages [3]. To perform successful keyword/URL filtering, for any intercepted network flow a DPI tool will need to defragment all the packets belonging to that flow (or even across multiple flows), and

perform online/offline inspection of the defragmented content. This makes DPI-based keyword/URL filtering more resource-intensive compared to IP blacklisting and DNS interference specially given the sheer volume of traffic intercepted by typical monitoring devices.

CDN content is prone to the same level of keyword/URL filtering as regular (non-CDN) Internet content. In both cases, **the use of encryption (e.g., HTTPS) will effectively foil** any kind of keyword/URL filtering.

3.1.4 Self-censorship by CDN providers

State-level censors may require commercial CDN providers to self-censor forbidden Internet content or else they will not be able to run CDN infrastructure (e.g., edge servers) within the censors' network territories or make business with the publishers under the jurisdiction of the censors. Such self-censorship can be performed at different levels, from not hosting forbidden content publishers at all to not serving forbidden content to the censors' Internet users only. The larger a state-level censor's economy is, the higher the chances that commercial CDN providers will perform self-censorship.

3.1.5 Advanced resource-intensive analysis

In addition to searching for prohibited keywords or URLs, DPI tools can be leveraged for other types of analysis for the purposes of censorship, e.g., online/offline statistical traffic analysis [19] and protocol identification analysis [22, 24]. Such advanced types of analysis are significantly resource-intensive and suffer from high rates of false positives, which is why there is no evidence that real-world censors use such techniques at large-scale.

3.2 How China is Censoring CDN Content

We study how China's Great Firewall (GFW), unanimously believed to be the most competent and elaborated censorship system in the world [30,37], blocks CDN content. China also has the strongest economy and the most advanced IT infrastructure compared to other state-level censors. Given the exhaustive list of commercial CDN providers [8], we limit our study to the providers with the largest market share.

Experiment Methodology: We run our experiments on a Linux computer node located inside China (which we anonymize to protect the owners). We also have access to several computers outside China. We do verify that our node in China faces the same censorship restrictions as typical Chinese users: we try connecting to a list of blocked URLs known to be blocked based on `GreatFire.org` from this node, and we confirm that the URLs are inaccessible from our node.

We assemble a list of blocked CDN-hosted URLs by scrapping `GreatFire.org`, then, we analyze how they are blocked. To check if a target domain name faces DNS interference, we perform a `dig` query for its domain name and check whether the returned IP address corresponds to the queried domain name using an online `whois` service outside China. To check if a target URL is IP address filtered, we resolve the URL's domain name on a non-censored machine, and then try fetching the URL content directly from the Chinese node by directly connecting to the obtained IP address without performing a DNS lookup on the Chinese node. Our code that fetches a URL from its IP address with no DNS query is written in Python, and we refer to it as `GET(IP,URL)` in the rest of the paper. For an HTTPS URL, censors will not be able to see the domain name of the URL when `GET(IP,URL)` is used.

3.2.1 Akamai: non-Chinese CDN Present in China

Akamai is the leading CDN provider in the world with more than 170,000 edge servers (tripled since 2010 [34]) across 102 countries and within over 1,300 networks [1]. Among the major CDN providers we studied, Akamai is the only non-Chinese CDN provider with CDN infrastructure *inside China*, which is "offered through a partnership" as stated by the company [14].

To do our experiments, we assemble a list of China-based edge servers of Akamai by browsing several non-forbidden Akamai-hosted URLs from our Chinese node (Akamai's mapping system often returns an edge server close to the querying user). We verify that the edge servers are in China using an online whois service. We make the following observations:

Legitimate content can be retrieved from any edge server. For any of the experimented non-forbidden URLs hosted by Akamai, we are able to successfully fetch the URL using `GET(IP,URL)` from *any* of the Akamai edge servers, even those not returned by Akamai's mapping system to our DNS query.

China-based edge servers of Akamai censor queries for forbidden content. Contrary to what we observed for legitimate URLs, our Chinese node can fetch a forbidden Akamai URL *only* from the non-Chinese edge servers of Akamai. More specifically, our attempt to fetch a blocked URL from a Chinese Akamai edge server results in an "HTTP 403 Forbidden" error, which implies that the Chinese edge server "**can be reached and understood the request, but refuses to take any further action**."[2] This finding complies with the company's statement [14] that "because Akamai understands the business and institutional requirements involved in delivering your content within China, we can work with you to avoid any regulatory surprises."

The mapping system does not self-censor. Akamai's mapping system, which appears to be located outside China, does not perform any censorship on mapping requests from Chinese users, i.e., it returns them valid edge server IP addresses even when they ask for content forbidden by the GFW. An interesting case study demonstrating this is `istockphoto.com`: possibly due to an oversight, the website is blocked *only* using URL filtering, but neither DNS interference (a DNS query resolves correctly) nor IP blocking (the HTTPS version of the website is accessible). For this website, we observe that the DNS queries sent to Akamai's mapping server from our Chinese node are resolved to a valid Akamai edge server *outside China*, confirming that the mapping system does not self-censor, contrary to the China-based edge servers.

DNS interference is the most common technique practiced to block CDN content. The majority of the blocked CDN websites we examined are DNS poisoned by the GFW, i.e., our DNS queries are all resolved to invalid IP addresses.

No edge server IPs are blocked by the GFW , even the edge servers outside China. This confirms our hypothesis in Section 3.1.1: censors do not block edge server IPs because

[2] http://en.wikipedia.org/wiki/HTTP_403

they serve both legitimate and forbidden content, there is too many of them (170,000 Akamai edge servers), and their assignment is so dynamic that can not be mapped to specific content publishers.

DNS interference is used to block forbidden HTTPS URLs. While non-encrypted URLs are easily blockable by DPI, to block encrypted URLs the censors have no technical way other than blocking the whole domain serving the URL. This can be implemented either by DNS interference or IP address blocking, but we observe that DNS interference is the standard technique used by the GFW due to the discussed challenges with IP address blocking. This, however, may block some non-forbidden content as well, as observed previously [29]. For instance, to block `https://a248.e.akamai.net/f/1/1/1/www.psiphon.ca/zh/download.html`, a mirrored link to download Psiphon [27], the GFW has DNS poisoned (but not IP address filtered) the whole `a248.e.akamai.net` domain, which has also blocked some content of `ikyu.com`, a non-forbidden online travel service.

3.2.2 Non-Chinese CDNs not Present in China

We also studied CloudFlare, Amazon CloudFront, EdgeCast, Fastly, and SoftLayer, which are major CDN providers with the largest market share after Akamai, but with no CDN infrastructure inside China. Our experiments confirm online information[3,4,5] [9] that none of these providers have CDN infrastructure inside China, i.e., our requests for non-blocked websites from our Chinese node are always resolved to edge server IP addresses *outside China*. We make the following observations:

Content can be retrieved from any edge server of the CDN provider hosting a specific URL, even from the edge servers not assigned by the CDN mapping system.

We find no evidence of self-censorship by the studied CDN providers. Forbidden content can be successfully retrieved through all edge servers (using `GET(IP,URL)`) from our Chinese node.

DNS interference is the most common technique used by the GFW to block forbidden content hosted by these CDN providers. Once our Chinese node sends a DNS query for a forbidden CDN hosted content, it receives a non-relevant IP address (instead of a valid edge server address).

We observe no edge server being IP filtered by the GFW. This is due to the collateral damage caused by blocking non-forbidden content, as discussed in Section 3.1.1.

HTTPS URLs are censored through DNS interference. Blocking encrypted CDN-hosted URLs is not feasible through IP address blocking (due to the use of shared edge server IPs) nor DPI (due to encryption). To block, the GFW interferes with the name resolution of forbidden encrypted CDN URLs, which can results in blocking some non-forbidden content.We noticed that many of the URLs blocked by the GFW are the mirror webpages set up by `greatfire.org`'s "collateral freedom" project [16], e.g., `https://daol8yb47gnd3.cloudfront.net`. We also observe that the GFW has DNS poisoned (but not IP filtered) some CDN domain names because the domain names were used by some forbidden content publishers. For example, to block `https://edgecastcdn.net/00107ED/freeweibo` the GFW is DNS poisoning the whole `edgecastcdn.net` domain. In November 2014, such DNS poisoning accidentally blocked major non-forbidden CDN customers in China, including the HSBC's banking portal website.[6]

3.2.3 Chinese CDN Providers

ChinaCache, ChinaNetCenter, and CDNetworks are the leading [9] Chinese CDN providers with ChinaCache providing the fastest performance thanks to its 11,000 servers. All three CDN providers fully comply with the regulations and laws of the Chinese government. Not surprisingly, we could not find *any* censorship by the GFW on the content hosted by these CDN providers, presumably because they do not host any forbidden content.

3.3 Lessons of Our Study

Table 1 summarizes our analysis, discussed below:

1. End-users can bypass CDN mapping and directly fetch CDN content from *any* of the edge servers of the hosting CDN provider. The mapping system of a CDN provider dynamically returns the edge servers that provide end-users with the best quality-of-service based on factors like end-user's geographic location, network conditions, etc., however, this does not prevent end-users from directly fetching content from the other edge servers of the CDN provider. Therefore, if an end-user knows *some* edge servers of the hosting CDN provider, she can ignore the name resolution (CDN mapping) step —which may be interfered with by the censors —and directly fetch content from the known edge servers. This may slightly downgrade the quality-of-service to the user as evaluated in Section 5.3. Also, if some (but not all) of the edge servers perform self-censorship (e.g., Akamai in China), end-users can still get the content from the other (non-censoring) edge servers of that CDN provider.

2. The GFW avoids IP address filtering in blocking CDN content because of its significant collateral damage, i.e., blocking (popular) non-forbidden content. That is, IP address blocking of forbidden websites hosted on a CDN network will disable *all* the legitimate websites hosted by the same CDN provider as well. Even for forbidden websites hosted on P-CDN networks, IP address blocking will be exhaustive due to the large number of IP addresses in any single P-CDN network.

3. DNS interference is the GFW's main technique in censoring CDN content. Therefore, if end-users are able to obtain the IP addresses of the hosting edge servers from means other than DNS name resolution **they will be able to circumvent censorship** of CDN content.

4. To be allowed to run CDN infrastructure in China CDN providers must self-censor. Chinese CDN providers fully comply with the GFW regulations by not hosting any forbidden content at the first place. Non-Chinese CDN providers have to perform partial self-censorship to be allowed to run servers in China. For example, we found that Akamai is censoring forbidden content on its Chinese edge servers (but not other edge servers or its mapping system). *This could be the main reason why many major CDN providers, like Amazon CloudFront, are hesitant to run CDN infras-*

[3] `www.cloudflarestatus.com`
[4] `http://aws.amazon.com/cloudfront/details/`
[5] `knowledgelayer.softlayer.com/faqs/213`
[6] `http://www.theguardian.com/world/2014/nov/18/china-blocks-hsbc-web-crackdown-censorship`

Table 1: How the GFW censors content hosted by major CDN providers.

CDN provider	Infrastructure in China?	Self-Censorship?	DNS Poisoning Publisher Domains?	DNS Poisoning CDN Domains?	IP Filtering Edge Servers?
Non-Chinese, present in China (Akamai)	Yes	Yes, Chinese edge servers only	Yes, pervasively	Limited	No
Non-Chinese, not present in China (CloudFlare, CloudFront, EdgeCast, Fastly, SoftLayer)	No	None	Yes, pervasively	Limited	No
Chinese CDN (ChinaCache, ChinaNetCenter, CDNetworks)	Yes	Yes, entirely	No	No	No

tructure in China despite the significant economic attraction.

5. Non-Chinese CDN providers are constrained in how much they can self-censor. This is due to two main reasons: First, CDN providers have to abide by the regulations and laws of their home countries, which could deprive them from censorship. For instance, U.S.-based CDN providers cannot deny service to legitimate U.S. content publishers, including the ones forbidden by a foreign state-level censor like China. Second, excessive self-censorship can seriously hurt their reputation, and therefore their business interests. As witnessed, Akamai only partially cooperates in censorship with China.

6. Blocking encrypted CDN content is technically impossible without cooperation of the hosting CDN provider. Therefore, as long as there are some (shared) CDN providers not cooperating with the censors, content publishers can use *them* to resist censorship of their content.

4. CacheBrowser: A PUBLISHER-CENTRIC CIRCUMVENTION SYSTEM

Content caching by CDN providers has become an essential, inseparable part of Internet communications. However, our analysis summarized in Section 3.3 shows that *the censors face significant technical and non-technical challenges in blocking forbidden CDN content*. This motivated us to design *CacheBrowser*, a circumvention system that leverages the censors' challenges in blocking CDN content.

The high-level intuition behind CacheBrowser is that *the main technique used (and the only one feasible) to censor encrypted CDN content is interfering with name resolution, however, name resolution is not required to retrieve CDN content*, but it only aims at optimizing CDN communications. Therefore, a (censored) end-user can bypass name resolution and retrieve content from *any* of the known edge servers part of the hosting CDN provider, potentially with degraded quality-of-service as evaluated in Section 5.3.

Note that, this is only effective in circumventing CDN censorship: regular, non-CDN content can be mapped to a small, usually static, set of IP addresses, which can be easily IP filtered by the censors with no collateral damage (this is how Tor relays are blocked for instance). IP filtering CDN content, however, will cause the censors significant collateral damage since (as discussed in Section 3.1.1) CDN IP addresses are shared within a massive pool of heterogeneous, unrelated content publishers, most of them may be hosting non-forbidden (and possibly popular) content. Even for private CDN networks, IP filtering will be exhaustive to the censors as it will require them to block hundreds to thousands of IP addresses *per forbidden content publisher*, as opposed to a handful of IP addresses for non-CDN content.

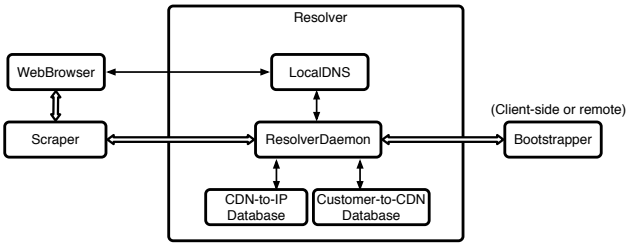


Figure 1: Block diagram of CacheBrowser.

CacheBrowser takes a unique, potentially-revolutionary approach for circumvention different from that of traditional circumvention systems like Tor [18], Anonymizer [5], and Psiphon [27]. In this approach, which we call *publisher-centric circumvention*, each content publisher plays a key role in having *her content* unblockable. This is unlike proxy-based circumvention systems like Tor in which case third-party entities (e.g., volunteer Tor relays) are the center of circumvention, and content publishers are mainly oblivious to the whole circumvention process. In Section 7.1.2 we enumerate the unique advantages of a publisher-centric approach like CacheBrowser over proxy-based systems.

4.1 CacheBrowser's Architecture

Figure 1 shows the block diagram of CacheBrowser's client software, which is installed on an end-user's machine. `WebBrowser` is a regular (possibly modified) web browser used to browse both regular and blocked Internet content. For any URL requested by the end-user or part of a previously fetched webpage, `WebBrowser` queries a local name resolution system, `LocalDNS`, to obtain the hosting IP address, and it contacts a regular DNS system *only* if there is no record in `LocalDNS` for the requested domain name.

`Scraper` continuously inspects `WebBrowser`'s requested URLs (including URLs entered by the user and the inner URLs of downloaded pages) looking for potentially blocked domain names. `Scraper` will ask `Resolver` to resolve newly-found blocked domain names, and the `Resolver` will resolve the blocked domain name and will add a record for it in `LocalDNS`. The `WebBrowser`'s DNS cache is then flushed to remove the existing DNS records for the blocked domain. This can be done either by restarting the browser

or through browser settings.[7,8] In case Resolver can not resolve a blocked domain (e.g., it does not know which CDN provider hosts a queried customer domain), it will ask a Bootstrapper for help. This will be done *only once* for each newly-found blocked domain name. The Bootstrapper component can be implemented either at the client-side or on a remote bootstrapping node, as will be described later.

4.1.1 *The* Scraper *Component*

The Scraper component has access to the web content of WebBrowser, e.g., as a browser plugin. It maintains a list of previously-resolved blocked domain names, ResolvedList, which will continuously update as the user browses new blocked websites. Suppose that a requested URL (either entered by the user or part of a downloaded HTML page) can not be successfully downloaded by WebBrowser (e.g., times out). In this case, if the URL's domain name is not in ResolvedList, Scraper will inform Resolver of the URL's domain name as a new domain name that should be resolved, and will add the domain name to ResolvedList. Otherwise, Scraper will inform Resolver that the current mapping for the URL's domain name in LocalDNS is invalid and should be updated.

If a requested URL is browsed successfully by WebBrowser, Scraper will report its uncensored domain name to Resolver along with its resolved IP address. Resolver may use this information to expand its CDN-to-IP database.

4.1.2 *The* Resolver *Component*

The Resolver is composed of four main parts, as shown in Figure 1: The LocalDNS database serves as a local DNS by keeping name records (i.e., DNS responses) for blocked domain names. ResolverDaemon is the engine of Resolver that processes queries from Scraper by accessing the other three databases of Resolver, or by communicating with the Bootstrapper component. The CDN-to-IP database contains a list of (presumably) unblocked IP addresses for each CDN domain. And, the Customer-to-CDN database records the mapping between each customer domain name and the domain name of its hosting CDN.

Adding a new LocalDNS record: Suppose that Scraper asks for a new domain name BlockedDomain to be resolved, which could be either a blocked customer domain name (e.g., www.cnn.com), or a blocked CDN domain name (e.g., a248.e.akamai.net). If Customer-to-CDN and/or CDN-to-IP have records for BlockedDomain, ResolverDaemon will add a corresponding record to LocalDNS. Otherwise, ResolverDaemon will query Bootstrapper to resolve BlockedDomain. If BlockedDomain is a customer domain (e.g., cnn.com), Bootstrapper will return BlockedDomain's CDN domain name(s) along with a few unblocked IP addresses of the hosting CDN domains. If BlockedDomain is a CDN domain, Bootstrapper will return several unblocked IP addresses of the CDN provider. ResolverDaemon will update the Customer-to-CDN and CDN-to-IP databases accordingly.

If a requested URL's BlockedDomain is a non-CDN domain but the inner URLs are CDN objects, an advanced Bootstrapper (as implemented in Section 5.1) may be used to return the non-CDN parts of the requested URL (e.g., a small-sized HTML page) and CacheBrowser will be able to fetch the (bulky) CDN parts of the URL.

Updating an existing LocalDNS record: If Scraper informs ResolverDaemon that an existing name resolution record for BlockedDomain needs to be updated, ResolverDaemon first removes the corresponding record from LocalDNS. If BlockedDomain is listed as a customer in Customer-to-CDN, ResolverDaemon checks if BlockedDomain's CDN provider has changed and updates Customer-to-CDN accordingly. Otherwise, ResolverDaemon will remove the unreachable IP address from CDN-to-IP database. Finally, it adds a new record for BlockedDomain in LocalDNS.

Updating other databases: ResolverDaemon continuously updates Customer-to-CDN and CDN-to-IP using Bootstrapper responses and other sources. For instance, when the user browses non-blocked CDN websites, ResolverDaemon will learn IP addresses of the hosting CDN domains, and use that information to update CDN-to-IP. Also, ResolverDaemon can download and import external Customer-to-CDN databases containing CDN mapping for popular blocked webpages through out-of-band channels.

4.1.3 *The* Bootstrapper *Component*

The *very first time* CacheBrowser tries to browse a blocked URL on a specific BlockedDomain, it is likely that CacheBrowser's Resolver can not resolve BlockedDomain, therefore, it will ask Bootstrapper for help. We assume that Bootstrapper has access to a non-censored name resolution oracle, and it can be implemented either at the client-side or as a remote service, as demonstrated in Section 5.1.

If the queried BlockedDomain is a CDN domain (e.g., a248.e.akamai.net) Bootstrapper will return to Resolver multiple IP addresses belonging to that domain. If BlockedDomain is a customer domain (e.g., cnn.com) Bootstrapper will return the name(s) of CDN domains hosting BlockedDomain's content, along with multiple IP addresses of those CDN domain(s). An advanced Bootstrapper may additionally download and send back the non-CDN parts of a requested URL (e.g., the base HTML page), which is implemented in our remote Bootstrapper in Section 5.1.

It is worth emphasizing that Bootstrapper is a *latency-insensitive, low-bandwidth* channel accessed *very infrequently* by a CacheBrowser client.

4.2 What Can Be Browsed by CacheBrowser

In the following we discuss how different classes of censored websites can be browsed by CacheBrowser.

Websites completely hosted on CDN. The majority of the CDN customer websites we analyzed host *all* of their web objects (e.g, HTML, CSS, JavaScript, and images) on CDN. Some customers host all of their objects on a single CDN system (e.g., https://www.istockphoto.com/) and some customers spread their objects across multiple CDN systems (e.g., https://www.pinterest.com/). Such websites can be browsed by CacheBrowser if at least some of the edge servers of their hosting CDN provider(s) serve the content to the public with no self-censorship based on users' geographic locations. This includes all of the non-Chinese CDN providers studied in Section 3 (including Akamai) with respect to the GFW. Also, the hosting CDN provider should encrypt connections to prevent the censors from keyword filtering. Fortunately, a large fraction of today's websites already satisfy both of the conditions: an increasing number of

[7] http://www.kahunaburger.com/2009/03/18/clear-dns-cache-in-firefox/

[8] http://superuser.com/questions/203674/how-to-clear-flush-the-dns-cache-in-google-chrome

content publishers are moving their services to commercial CDN networks (for purposes other than circumvention, e.g., improving end-user QoS and reliability), and all major CDN providers support (and some mandate[9]) encryption of connections. Therefore, CacheBrowser can already be used to browse a comprehensive list of blocked Internet content with no action needed from their publishers. Content hosted on P-CDN networks can be IP filtered if the size of the private CDN network is not prohibitively large.

Websites partially hosted on CDN. Some websites host only parts of their web objects (e.g., images and other heavy-weight objects) on CDN and the rest of the (light-weight) objects on their origin servers. This appears to be on outdated practice, and most recent CDN implementations host the entire website on CDN. For such a website, CacheBrowser can be used to download the CDN-hosted objects of the website, however, the non-CDN objects should be retrieved through other means. Since the non-CDN objects are usually light-weight (e.g., the base HTML page), a low-capacity covert channel can be used for retrieving them. In our current implementation of CacheBrowser presented in Section 5, we use our light-weight remote `Bootstrapper` for this purpose.

Websites not hosted on CDN. The publishers of such Websites can make their content browsable through CacheBrowser by simply hosting their content on a public CDN network that supports encryption. Thanks to the business competition between numerous commercial CDN providers, offering low-cost or even free [13] CDN service, CDN hosting has become affordable even to host personal websites [10] and blogs [12]. In particular, some providers offer free CDN hosting services [13], including CloudFlare, Incapsula, CoralCDN, and SwarmCDN. Hosting a website on a commercial CDN network requires no technical knowledge and is seamless: it can be done manually [12] by changing the DNS setting of the web host to the name servers of the CDN provider, and even many commercial web hosting services have integrated, one-click mechanisms to host a customer website on the commercial CDN network chosen by the publisher.[10]

Also, publishers can seamlessly migrate their CDN-hosted websites to other CDN networks in case of self-censorship by their current CDN provider; this can be done by updating the DNS settings of the web hosting system and usually takes less than 24 hours for the new DNS records to propagate. To improve resistance, a content publisher may even decide to host its content on multiple, competing CDN systems. Note that this does not substantially increase the costs to the publisher as (non-free) CDN providers charge per traffic volume served to end-users.[11]

5. IMPLEMENTATION AND EXPERIMENTS

We have built an implementation of CacheBrowser in Linux, and have evaluated its performance on our China-based node introduced in Section 3, which is censored by the GFW, and several other non-censored nodes.

[9] https://blog.cloudflare.com/introducing-universal-ssl/
[10] https://www.cloudflare.com/hosting-partners
[11] http://www.cdn.net/pricing/

5.1 Implementation

CacheBrowser's client software We have implemented a CacheBrowser client software in Linux, but the techniques can be easily implemented in other operating systems like Microsoft Windows. Our `WebBrowser` is a regular Mozilla Firefox browser. We have implemented the `Scraper` and `ResolverDaemon` components in Python, and we use two locally stored "txt" files for the `Customer-to-CDN` and `CDN-to-IP` databases. We simply use the "/etc/hosts" file of Linux OS to serve as our `LocalDNS` database.

In our prototype implementation, for any URL unreachable (either fully or partially) by `WebBrowser`, the user will have to run the `Scraper` Python script for that URL *only once*, and then re-try to fetch the URL using `WebBrowser`. Our `Scraper` will trigger the `ResolverDaemon` script, which, if successful, will add a DNS record for the blocked domain name(s) to the "/etc/hosts" file. In our final code release, we plan on building a Firefox plugin that automatically launches the `Scraper` script for each timed out URL.

We also implement a light-weight `WebBrowser` in Python for large-scale, scripted measurement of download latencies. For a given blocked URL, `https://blocked.com/BlockedURL.html`, our script obtains an edge server IP from "/etc/hosts" and sends an HTTP GET request to it for `BlockedURL.html` with the HTTP header fields of `Host: blocked.com/` and `User-Agent: Mozilla/5.0 (Windows NT 5.1; rv:14.0) Gecko/20100101 Firefox/14.0.1`.

Client-side Bootstrapper We prototype a client-side `Bootstrapper` using a free online DNS resolvers, `digwebinterface.com`. Such online DNS resolvers are free, non-blocked, and are able to answer DNS queries on behalf of multiple, geographically-dispersed DNS servers in different network regions.

Remote Bootstrapper We have also implemented a remote `Bootstrapper` based on SWEET [39], a light-weight covert communication system designed by our group previously. SWEET works by encapsulating covert messages inside emails sent using regular email protocols. We have implemented a SWEET server on a Linux server in Amherst, which listens for incoming emails from CacheBrowser clients targeted to a specific email address. Each incoming email contains one or multiple DNS queries for blocked URLs/domains generated by `ResolverDaemon` of CacheBrowser clients. We run a Python script on our `Bootstrapper` server that automatically processes such DNS queries, and sends the DNS responses back to the querying CacheBrowser clients. To answer a query, the Python script simply runs `dig` commands on 20 dispersed PlanetLab [15] nodes, or uses a local directory of domains resolved in the past hour. Our SWEET-based `Bootstrapper` is also able to download and send to CacheBrowser users the non-CDN parts of their queried URLs (e.g., a non-CDN HTML file).

5.2 Coverage Evaluation

As discussed in Section 4.2, CacheBrowser can be used to unblock censored content hosted on a CDN provider supporting HTTPS (CacheBrowser may be able to unblock some non-CDN or non-encrypted content as well, but the censors can eventually re-block them through IP address filtering or DPI). This already comprises a large fraction of today's Internet content, which is rapidly expanding with CDN hosting becoming more adopted by content publish-

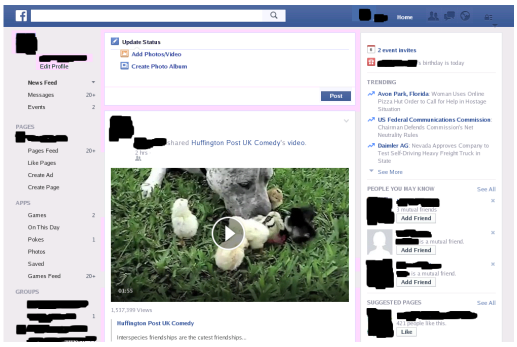


Figure 2: A snapshot of unblocking Facebook in China using CacheBrowser, with all objects (hosted on various CDN domains) fetched successfully.

ers. To demonstrate this, we scraped `greatfire.org` for websites in the top 1000 Alexa websites that are blocked by the GFW. We observe that 82% of the blocked websites are already hosted by some CDN provider, where the number is 85% for news websites like the `wsj.com`. As discussed in Section 4.2, non-supported publishers can take simple measures to make their content unblocked by CacheBrowser.

While the majority of these websites are hosted solely on commercial CDN networks, some have complex implementations by combining private and shared CDN networks. Facebook.com has by far the most complex design: a private CDN, `*.facebook.com`, hosts the base HTML pages including Facebook's frontpage and login HTML pages, and `*.akamaihd.net`, an Akamai S-CDN, serves many of the inner links of a Facebook session including all the CSS, JavaScript, and gif resources, as well as profile and non-profile images in the jpeg and png formats. We also notice that recently uploaded images and videos are served on `*.fbcdn.net`, another private CDN, before they are moved to `*.akamaihd.net`.

We find that the GFW performs DNS poisoning for the `*.facebook.com` domain *only*, but not the other domains as this effectively makes `facebook.com` inaccessible to the Chinese users. In addition to DNS poisoning, the GFW performs limited IP filtering for the `*.facebook.com` private CDN network, but still missing many of the IP addresses in this private CDN. As with other studied websites, we observed no instance of IP filtering for any of the shared CDN networks. All of the `*.akamaihd.net` domains perform self-censorship as discussed before, e.g., the edge servers located inside China deny to serve the content (but not the other edge servers).

Our CacheBrowser client is able to successfully unblock `facebook.com` (in addition to many other experimented blocked websites), which is a highly complex, multi-CDN webpage. As the snapshot in Figure 2 shows, our CacheBrowser user in China fetches all the objects successfully.

5.3 Latency Evaluation

As discussed earlier, the goal of a CDN network's mapping system is to maximize the QoS (e.g., reduce latency) for end-users by returning edge server IP addresses tailored to each querying end-user, network conditions, etc. CacheBrowser bypasses such mapping —which is already DNS poisoned by the censors —and uses *some* known edge server to fetch blocked CDN content. Therefore, one can expect that fetching content from an "alternative" edge server, as opposed to the "default" one mapped by the CDN mapping system, will reduce QoS for a CacheBrowser user.

We have measured such increased latency for three major CDN providers, as shown in Figure 3. The most left bar in each figure is the download time when the object is retrieved from the default edge server returned by the mapping system, and all the other bars correspond to various alternative edge servers of the same CDN network (each bar shows the mean over 100 trials). We have selected alternative edge servers that are highly distributed across the world, from Japan and Australia to the U.S., Europe, and the Middle East. As can be seen, while the latency increases when content is fetched from alternative edge servers, the overhead is not prohibitively significant. This extra latency is the price paid by CacheBrowser for unblocking the content that is otherwise unbrowsable. Also, our measurements reflect *the most pessimistic overheads* since we have chosen alternative edge servers that are far distanced from our end-user; a more strategic selection of edge servers by CacheBrowser (e.g., by choosing nearby edge servers) will reduce the latency overhead.

Figure 4 compares the download times of several CDN webpages by CacheBrowser, Tor, and a non-censored connection. We perform two sets of experiments, one from our China-based CacheBrowser client, and one from a local CacheBrowser client in our lab. In both of the cases, CacheBrowser results in higher latencies compared to a non-censored download, however, the overhead is not significant. Despite the increased latency, CacheBrowser performs significantly faster than Tor on our local client. We were not allowed to run Tor on our Chinese client, however, we conjecture a similar latency advantage over Tor in China. This is due to CacheBrowser's publisher-centric approach, in which content is not relayed through (potentially overloaded and resource-limited) third-party proxies that add extra delay.

5.4 Remote Bootstrapper

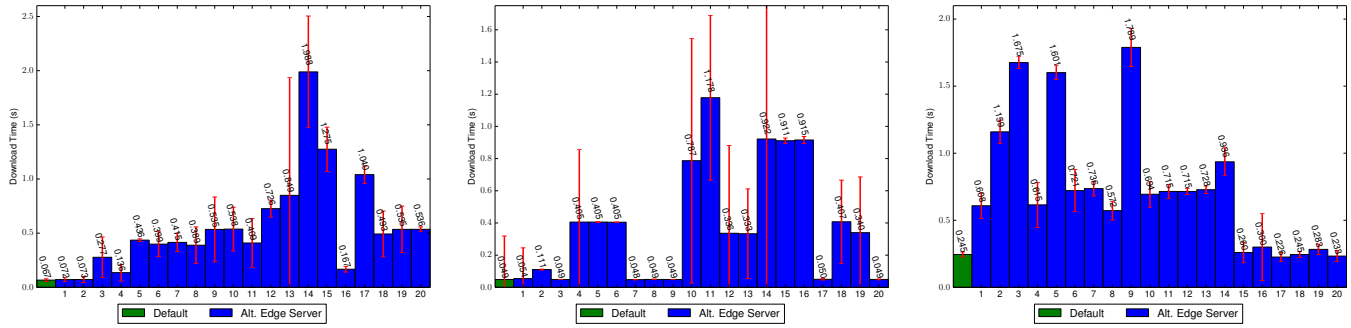
In our experiments, a CacheBrowser client receives an email containing DNS responses from our SWEET-based remote `Bootstrapper` with a median latency of 5.4 seconds (95% of messages are received within 10 seconds across 100 runs, confirming our previous study [39]) if the DNS responses are cached on the server. Such latency depends on factors like the client's distance from the SWEET email server and the client's email provider [39]. If the DNS response is not cached on the remote `Bootstrapper` server, running `dig` commands on PlanetLab nodes will add another 3-8 seconds delay, depending on the responsiveness of the used PlanetLab nodes.

Such latency is good enough for CacheBrowser to operate because a typical content publisher does not change its CDN provider frequently (e.g., only every few months), therefore, a CacheBrowser client needs to re-query a blocked content publisher *only once* after every migration.

6. SYSTEM ANALYSIS

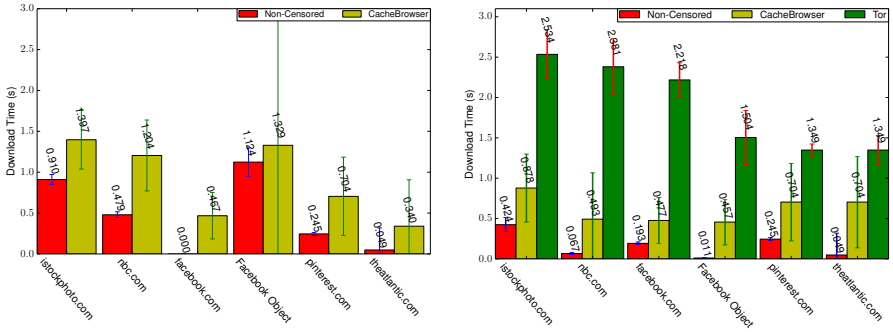
6.1 Threat Model

We assume that our CacheBrowser user resides inside a censored network territory. The censoring ISP may deploy various censorship mechanisms as discussed in Section 3.1



(a) Akamai (b) EdgeCast (c) Amazon CloudFront

Figure 3: Comparing download time of a CDN content object from the "best" edge server returned by mapping system vs. other edge servers. The alternative edge servers are chosen to be geographically far distanced from the end-user.



(a) Client in China (b) Client in Amherst, MA, U.S.

Figure 4: Comparing download latency for several websites using three methods: non-censored (regular) download, using CacheBrowser, and using Tor. We were not allowed to run Tor on our Chinese client. Also, in our China experiments, there is no "Non-censored" measurement for `facebook.com`, which is blocked, and we use the HTTPS version of `istockphoto.com` for its "Non-censored" measurement (it is blocked by keyword filtering only). All the other websites are not blocked in China.

in order to block forbidden content as well as to detect and disable the use of any censorship circumvention system like CacheBrowser. However, we assume that the censors are rational, i.e., they refrain from any actions that will interfere with non-prohibited Internet activities of a significant number of their non-circumvention citizens. In particular, we assume that the censors do not block *all* encrypted traffic as encryption is essential for various popular non-forbidden Internet services. Additionally, we assume that the censors do not entirely block a commercial CDN provider (e.g., by IP blacklisting all edge servers) merely because it serves some prohibited content publishers, since the CDN provider will likely be hosting many non-forbidden content publishers as well. The censors, however, may perform —selective —blocking of CDN domain names and encrypted traffic.

We assume that commercial CDN providers do not cooperate with censored users nor with content publishers in order to circumvent censorship, as this may jeopardize their business interests with economically-powerful state-level censors like China. A CDN provider may cooperate with the censors, however, the cooperation is constrained to not violate the jurisdiction of the CDN's home country as well as its business interests in other parts of the world. For instance, as we analyzed in Section 3, Akamai only partially cooperates with the GFW in order to protect its business interests and reputation in other (non-censored) regions. We assume that the CDN providers fully controlled by the censors (e.g., Chinese CDN providers) will not even host any forbidden content.

In this paper, we do not consider unobservability against active attacks and traffic analysis, but discuss the challenges and possible solutions in the rest of this section.

6.2 Privacy

Privacy form Circumvention Provider Proxy-based circumvention systems like Tor, Psiphon, and VPNs expose their users to significant privacy risks from the circumvention providers. For instance, malicious Tor relays can perform a toolset of attacks [31] to comprise Tor users' privacy, and a malicious VPN service (e.g., one run by a repressive government to spy on dissidents) can learn the users' browsing activities and even the content of their communications. Such risks do not apply to CacheBrowser due to its publisher-centric approach, i.e., no proxy is used.

A CacheBrowser client may use a remote `Bootstrapper`, e.g., the email-based system described earlier. The remote `Bootstrapper` can not see the content of the client's communications, but may learn the destinations she browses. Even

this is not a risk, because the `Bootstrapper` will only see the user's (anonymized) email address, not her IP address.

Privacy from CDN providers The CDN provider hosting a blocked website will be able to identify the censored users who fetch the blocked website, e.g., based on IP address information. This does not introduce a new privacy risk: CDN providers can anyway identify all of the end-users browsing their customers (including both censored and uncensored end-users), however, as part of their service agreement they guarantee to not misuse such information, e.g., not disclose it to state-level censors. A CDN provider releasing such information will risk hurting its reputation and losing its customers.

Privacy from the censors Suppose that a CacheBrowser user downloads an encrypted blocked CDN content, e.g., an HTTPS webpage. Unless the CDN network hosting the blocked content cooperates with the censors, the censors will not be able to detect neither the fact that the client is browsing that blocked website, nor the content of her communication. This is because the hosting edge servers serve many other webpages, CacheBrowser bypasses the DNS stage, and encryption hides the actual HTTP GET requests and responses. If no encryption is used, the censors can detect such communication using DPI and block it.

6.3 Blocking Resistance

Censors disabling encryption Outright blocking of encryption will prevent CacheBrowser users from hiding the web destinations they browse. However, as stated in our threat model, we assume that the censors will refrain from doing so due to the significant collateral damage of disabling legitimate, critical Internet services that use encryption.

Censors blocking CDN networks Censors may decide to block a whole CDN domain (e.g., `a248.e.akamai.net`) that hosts some forbidden CDN customers. The censors can do so by DNS interference, however, this will not impact CacheBrowser users as they directly connect to edge servers. The censors can alternatively block *all* IP addresses of the target CDN domain. However, as discussed before, this will cause the censors significant collateral damage due to blocking the non-forbidden websites served through those IP addresses. Also, enumerating and blocking *all* of the IP addresses of a CDN domain (which also may change over time) will be exhaustive to the censors.

CDN provider cooperating with the censors In order to protect their business interests, some CDN providers may decide to cooperate with the economically strong censors. As discussed before, such cooperation will be limited as the CDN providers have to protect their reputation and business interests in other regions. For instance, CDN providers based in the U.S. are not allowed to deny service to particular content publishers whose content is legitimate in the U.S. (but might be forbidden in a censoring region like China). This is confirmed by our analysis of Section 3.3, i.e., Akamai, the only non-Chinese CDN cooperating with the GFW, performs its self-censorship in a limited manner. Also, CDN cooperation with censoring countries with smaller economic impact (e.g., Syria) seems to be very unlikely.

Denial-of-service attacks Censors occasionally lunch denial-of-service (DoS) attacks on circumvention systems to interfere with their operation. This is usually performed by flooding the third-party proxies running a circumvention system, e.g., by building long Tor circuits to exhaust Tor relays [20]. Such DoS attacks are out of the scope for CacheBrowser as it does not involve any third-party proxies.

Active attacks Previous work [22, 24] shows how active attacks, e.g., strategically dropping packets, can be used to identify and block some circumvention systems. Such attacks do not generally apply to CacheBrowser's encrypted HTTP traffic, e.g., the dropped packets will be resent at the TCP layer. Nonetheless, the dependencies between multiple HTML objects in a complex webpage (or across multiple webpages) may be used to devise attacks specific to that webpage. We leave a thorough analysis to future work.

Insider attacks Unlike traditional circumvention systems, censors who run CacheBrowser do not learn additional information that will help them in blocking CacheBrowser. For instance, in proxy-based circumvention systems like Tor censors can learn (and block) the IP addresses of circumvention proxies by merely running such systems [37]. Also, our remote `Bootstrapper` sends various DNS responses to querying users to be unobservable to insider attackers.

Reconfiguring the GFW to block CacheBrowser CacheBrowser does not rely on an existing flaw in the GFW nor a misconfiguration of the censoring firewalls that can be fixed once/if CacheBrowser has become widely deployed and popular. Instead, it relies on the censors' technical difficulties in distinguishing forbidden CDN content from benign CDN content, and also the challenges of blocking forbidden CDN content without significant collateral damage.

6.4 Website Fingerprinting

Website fingerprinting can compromise both privacy and blocking resistance, using different techniques:

Based on traffic patterns The censors may use statistical fingerprinting techniques [7, 23] on encrypted CDN traffic to detect end-users connecting to forbidden CDN customers. Such techniques are resource-intensive to be performed at large-scale and suffer from high rates of false positives (thus, collateral damage) as demonstrated by a recent study [28]; this is why there is no evidence of real-world deployment of such techniques by the censors. If such techniques are deployed content publishers of our publisher-centric circumvention system can increase the rates of false positive even further by dynamically changing their patterns or resembling non-forbidden pages.

Based on webpage objects A typical webpage may contain URI links to various external web resources such as HTML CSS, analytics JavaScript, and advertisement. Even though some of these external resources are fetched through encrypted connections, one might fingerprint a webpage based on the number, type, and destination of its external URIs. As a future work, we plan to explore the extent of such website fingerprinting on CacheBrowser connections, and devise countermeasures mechanisms. Possible countermeasures include blocking unnecessary external resources, pre-fetching required external resources, and fetching external resources that mimic other, non-blocked webpages.

Based on edge servers At any given time, a CDN's mapping system is likely to return different edge servers to a user for two different customer webpages. This can be used to probabilistically fingerprint CDN webpages despite encryption. A simple countermeasure is for a CacheBrowser client to always connect to edge servers that are returned for some

non-forbidden webpage on the same CDN system. We leave further analysis and implementation to future work.

6.5 Availability of Remote Bootstrapper

CacheBrowser's remote `Bootstrapper` may be targeted by DoS attacks, but the risks are very limited due to the low volume of traffic and light weight of operations in each `Bootstrapper` communication. Also, a `Bootstrapper` can use standard DoS defense mechanisms like using computational puzzles as suggested for SWEET [39].

The utilized remote `Boostrapper` should provide plausible unobservability, otherwise it will be blocked by the censors. The remote `Boostrapper` of our current implementation is based on SWEET [39], whose unobservability has been extensively discussed in its original paper [39]. SWEET offers high unobservability for low-volume communications, making it an ideal choice for our light-weight remote `Bootstrapper`. We use our SWEET-based `Bootstrapper` to optionally download and send to CacheBrowser users the non-CDN parts of their queried URLs, as discussed before. This will increase our `Bootstrapper`'s traffic volume and weaken its unobservability.

It is worth emphasizing that CacheBrowser is not tied to the presented SWEET-based `Bootstrapper`, and future work can design alternative light-weight remote `Bootstrappers`, e.g., based on online social networks. Also, a remote `Bootstrapper` is not required for CacheBrowser's operation and a client can use client-side `Bootstrappers`.

6.6 Resistance to Nation-State Censors

The practical censorship resistance offered by CacheBrowser depends on specific nation-state censors. For each particular nation-state censor, `CensoringCountry`, we classify commercial CDN providers into three categories: (1) CDN providers that are not under the `CensoringCountry`'s jurisdiction *and* do not have any explicit business relations with them (e.g., Fastly in the case of China), (2) CDN providers that are not under `CensoringCountry`'s jurisdiction *but* partially cooperate with them to protect their business interests (e.g., Akamai in the case of China), and (3) CDN providers that are under the full jurisdiction of `CensoringCountry` (e.g., ChinaCache in the case of China). As discussed for China in Section 3.2, CacheBrowser can be used to browse forbidden content on the first and second categories of CDN providers as long as the hosting CDN provider does not perform full cooperation with the censors (e.g., through geo-location based content filtering, or removing forbidden content). Such censorship cooperation is less likely for CDN providers of type one due to the lack of business interests, but is more likely to be implemented by the second category of providers. The CDN providers of the third category simply do not host forbidden content.

Fortunately, for major nation-state censors the majority of commercial CDN providers fall under the first category of providers discussed above. As mentioned in Section 3.2, there is only one provider in the second category for China. The situation is even worse for nation-state censors with weaker economies, for instance we could not identify any commercial CDN providers that run edge servers in Iran, Syria, or Venezuela, i.e., their second category is empty. Even if some of these censors deploy their home-brewed CDN systems (i.e., the third category) they can not disconnect their users from the non-censoring CDN providers due to the significant collateral damage discussed earlier.

7. RELATED WORK

7.1 Proxy-Based Circumvention Systems

Proxy-based circumvention [5,6,18,25–27,38] is the mainstream approach for censorship circumvention. In this approach, one or multiple circumvention entities, known as proxies, relay traffic between censored users and content publishers. Such proxying is offered either as a paid or free service, and does not involve content publishers.

In CacheBrowser's publisher-centric approach, however, third-party involvement is minimal, and end-users obtain content directly from content publishers. Obviously, the way a publisher serves its content will impact the performance of a publisher-centric system.

7.1.1 *Domain Fronting*

Domain fronting [21] is a new technique for setting up proxies in proxy-based circumvention systems to make them better resistant to IP address blocking. The main idea of domain fronting is to run circumvention proxies, e.g., Tor bridges [17], on "web services" that share IP addresses with multiple (potentially non-related) guest services. This way, IP address blocking the circumvention proxy (e.g., the Tor bridge) running inside the web service will cause the censors collateral damage since blocking shared IP addresses will block all the other (potentially non-forbidden) guest services running on the hosting web service. The web services used for domain fronting include application engines, cloud computing infrastructures, and CDN systems.

Domain fronting has been recently adopted by several proxy-based circumvention systems, in particular Tor (as a pluggable transport [32]), Psiphon, FireFly Proxy,[12] and Flashlight HTTP proxy.[13] Tor's meek pluggable transport implements domain fronting on Google App Engine, Microsoft Azure cloud infrastructure, and CloudFront CDN.

While the meek project's implementation on CloudFront CDN shares conceptual similarities with CacheBrowser, we emphasize that the two approaches of "domain fronting" and "cache browsing" are **fundamentally different**. Domain fronting is the approach to *run circumvention proxies* on online web services so that a proxy's IP address is shared with other (non-forbidden) applications, therefore costly to be blocked. On the other hand, our cache browsing approach (implemented as CacheBrowser) is the idea of obtaining a censored content object directly from one of its many cached versions, *without using any circumvention proxies*. Therefore, domain fronting and cache browsing have different implementation domains. Domain fronting can be deployed on web services that share IP addresses like app engines, cloud computing systems, and CDN, and cache browsing can leverage content caching by any caching system including CDNs, cache networks [33, 36], and filesharing networks like BitTorrent. In other words, domain fronting is "service-centric" while our cache browsing approach is "content-centric."

[12] https://github.com/yinghuocho/firefly-proxy
[13] https://github.com/getlantern/flashlight

7.1.2 Advantages of CacheBrowser

In the following, we discuss the advantages of CacheBrowser's publisher-centric approach over proxy-based circumvention systems, particularly domain fronting.

Superior quality-of-service: Proxy-based circumvention systems like Tor are notorious for their poor QoS. This is due to their traffic indirection through network relays that usually have insufficient resources (e.g., bandwidth, CPU, etc.) to support large numbers of censored clients. A publisher-centric approach improves the QoS by eliminating such third-party relays and exchanging content directly between publishers and end-users. As illustrated in Figure 4, CacheBrowser significantly outperforms Tor's meek implementation.

Minimal expenses to third-parties: Proxy-based circumvention systems rely on either third-parties (e.g., volunteer Tor relays) or end-users (e.g., paid VPN services and Anonymizer) to pay the monetary expenses of running circumvention proxies. In a publisher-centric approach, however, each content publisher pays the expenses for having its own content un-censored. In CacheBrowser, in particular, the price is indirectly paid by content publishers who host their content on commercial CDN systems.

Censors' pressure has less impact: Proxy-based circumvention systems, including domain fronting services like meek, FireFly proxy, and Flashlight proxy, are online "services" that are *solely used for censorship circumvention*. Therefore, the censors can pressure their hosting web services to entirely remove them. For example, Apple Inc. has repeatedly removed anti-censorship applications from its China iOS App Store under the pressure of the Chinese government.[14] In CacheBrowser, however, even if a CDN provider takes down forbidden content within the censors' region, users can still obtain the cached content from other regions not impacted by censorship.

Eliminating legal consequences for third-party operators: Volunteer operators of proxy-based circumvention systems often face legal punishments for the illegal actions of (usually unknown) end-users. This has frequently been witnessed[15] for volunteer Tor relays whose relays have been used by unknown Tor users for illegal activities. Publisher-centric systems eliminate such risks as no third-parties are involved in content transmission, and censored users obtain content the same way as regular, non-censored users do.

Little reliance on third-parties to operate: Once the third-parties running a proxy-based circumvention system lose interest in running the system (e.g., due to economic or political reasons), the system will stop functioning. A publisher-centric system like CacheBrowser, however, is to the most part independent of third-party operators, and end-users will be able to fetch content as long as content publishers are willing to serve content.

7.1.3 Disadvantage of CacheBrowser

Should be adopted by content publishers: A publisher-centric solution will not be effective if its underlying concept (CDN hosting in the case of CacheBrowser) is not embraced by content publishers, *either intentionally or unintentionally*. In the case of CacheBrowser, many content publishers have already deployed the underlying concept for reasons other than censorship resistance, as discussed in Section 5.2.

[14] https://en.greatfire.org/blog/2013/oct/opendoor-shut-apples-chinese-app-store

[15] https://network23.org/blackoutaustria/2014/07/01/to-whom-it-may-concern-english-version/

7.2 Mirroring Censored Content

A recent alternative approach to proxy-based systems is to mirror censored content on web services that support domain fronting, without running any third-party proxies. For instance, the *Collateral Freedom project* [16] mirrors several popular censored websites on cloud-fronting platforms like Amazon EC2 and GitHub. Similar to domain fronting proxy-based systems, the mirrored content will make use of IP addresses that are shared with other, likely non-forbidden, Internet content, therefore any censorship will cause the censors collateral damage due to making the non-forbidden content inaccessible as well. For example, blocking a mirror of NYTimes on GitHub (e.g., https://github.com/greatfire/wiki/wiki/nyt) will require the censors to block all the other GitHub projects as well.

Note that the mirroring approach has major *limitations*. First, mirroring only works for static, non-interactive web traffic, e.g., static news articles, but can not be used for dynamic web content, e.g., social networking services like Facebook. Second, since the mirrors are set up manually by volunteers, they are limited to popular, highly-demanded censored content only (e.g., the Collateral Freedom project mirrors a very small set of websites). Third, the mirrored content is often hosted by third-party volunteers, not the actual content publishers. This may impact the integrity of the content, or even users' privacy. For instance, the GFW may host mirrors of blocked content that are modified. By contrast, CacheBrowser users obtain the content form the same channel as regular non-censored users. Fourth, some mirrored content may be blocked by DNS poisoning. For instance, the GFW blocks Collateral Freedom mirrors hosted on some CDN networks through DNS blocking, enforcing the Collateral Freedom project to change the mirrors' CDN domains frequently. In fact, we were able to use our CacheBrowser tool to successfully unblock the previously blocked mirrors of the Collateral Freedom project.

8. CONCLUSIONS

Through analysis and real-world experiments, we demonstrated that blocking cached CDN content poses unique technical and non-technical challenges to the censors. We designed and implemented a client-side circumvention system that exploits such challenges and is able to unblock censored CDN content in China with a superior QoS compared to proxy-based circumvention systems like Tor. The "cache browsing" idea of CacheBrowser can be applied for censorship circumvention in other cache-centric contexts like next-generation Internet architectures [33, 36] and filesharing overlay networks. The broader lesson of our study is that future circumvention technologies should embrace modern, emerging communication paradigms in their designs.

9. ACKNOWLEDGMENTS

We would like to thank our anonymous reviewers for their insightful comments. The research was supported in part by a 2015 Faculty Research Award from Google Inc.

10. REFERENCES

[1] Akamai Facts & Figures. http://www.akamai.com/html/about/facts_figures.html.

[2] A. Anand, F. Dogar, D. Han, B. Li, H. Lim, M. Machado, W. Wu, A. Akella, D. G. Andersen, J. W. Byers, et al. XIA: An Architecture for an Evolvable and Trustworthy Internet. In *HotNets*, 2011.

[3] D. Anderson. Splinternet Behind the Great Firewall of China. *Queue*, 10(11):40, 2012.

[4] Anonymous. The Collateral Damage of Internet Censorship by DNS Injection. *ACM SIGCOMM CCR*, 42(3), 2012.

[5] J. Boyan. The Anonymizer: Protecting User Privacy on the Web. *Computer-Mediated Communication Magazine*, 4(9), Sept. 1997.

[6] C. Brubaker, A. Houmansadr, and V. Shmatikov. CloudTransport: Using Cloud Storage for Censorship-Resistant Networking. In *PETS*, 2014.

[7] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson. Touching from a distance: Website fingerprinting attacks and defenses. In *CCS*, 2012.

[8] Latest List Of Vendors In The Content Delivery Ecosystem. http://blog.streamingmedia.com/2014/07/cdnvendors.html.

[9] Content Delivery Networks. http://www.cdnplanet.com/cdns/.

[10] How to Use CloudFlare to Speed Up Your Site. http://www.northernbellediaries.com/how-to-use-cloudflare-to-speed-up-your-site/.

[11] 5 Reasons to Implement a Content Delivery Network (CDN). http://blog.newrelic.com/2012/12/18/5-reasons-to-implement-a-cdn/.

[12] Tutorial: Creating a CDN for WordPress with CloudFront and S3. http://blog.celingest.com/en/2013/07/19/tutorial-creating-cdn-wordpress-cloudfront-s3/.

[13] 10 Free CDN Services to Speed Up WordPress. http://www.wpexplorer.com/free-cdn-services-for-wordpress/.

[14] Reaching China's Online Users. http://www.akamai.com/html/technology/china-content-delivery-network.html.

[15] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. Planetlab: an overlay testbed for broad-coverage services. *SIGCOMM CCR*, 33(3):3–12, 2003.

[16] Collateral Freedom. https://openitp.org/pdfs/CollateralFreedom.pdf, 2013.

[17] R. Dingledine and N. Mathewson. Design of a Blocking-Resistant Anonymity System. https://svn.torproject.org/svn/projects/design-paper/blocking.html.

[18] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-generation Onion Router. In *USENIX Security*, 2004.

[19] M. Dusi, M. Crotti, F. Gringoli, and L. Salgarelli. Tunnel hunter: Detecting application-layer tunnels with statistical fingerprinting. *Computer Networks*, 53(1):81–97, 2009.

[20] N. S. Evans, R. Dingledine, and C. Grothoff. A Practical Congestion Attack on Tor Using Long Paths. In *USENIX Security Symposium*, 2009.

[21] D. Fifield, C. Lan, R. Hynes, P. Wegmann, and V. Paxson. Blocking-resistant Communication through Domain Fronting. In *PETS*, 2015.

[22] J. Geddes, M. Schuchard, and N. Hopper. Cover Your ACKs: Pitfalls of Covert Channel Censorship Circumvention. In *CCS*, 2013.

[23] D. Herrmann, R. Wendolsky, and H. Federrath. Website fingerprinting: attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier. In *CCS*, 2009.

[24] A. Houmansadr, C. Brubaker, and V. Shmatikov. The Parrot is Dead: Observing Unobservable Network Communications. In *IEEE S&P*, 2013.

[25] A. Houmansadr, G. Nguyen, M. Caesar, and N. Borisov. Cirripede: Circumvention Infrastructure Using Router Redirection with Plausible Deniability. In *CCS*, 2011.

[26] A. Houmansadr, T. Riedl, N. Borisov, and A. Singer. I Want My Voice to Be Heard: IP over Voice-over-IP for Unobservable Censorship Circumvention. In *NDSS*, 2013.

[27] J. Jia and P. Smith. Psiphon: Analysis and Estimation. http://www.cdf.toronto.edu/~csc494h/reports/2004-fall/psiphon_ae.html, 2004.

[28] M. Juarez, S. Afroz, G. Acar, C. Diaz, and R. Greenstadt. A critical evaluation of website fingerprinting attacks. In *CCS*, 2014.

[29] K. Kathuria. Bypassing Internet Censorship for News Broadcasters. In *FOCI*, 2011.

[30] C. Leberknight, M. Chiang, H. Poor, and F. Wong. A Taxonomy of Internet Censorship and Anti-censorship. http://www.princeton.edu/~chiangm/anticensorship.pdf, 2010.

[31] Z. Ling, J. Luo, W. Yu, X. Fu, D. Xuan, and W. Jia. A new cell counter based attack against Tor. In *CCS*, 2009.

[32] meek Pluggable Transport. https://trac.torproject.org/projects/tor/wiki/doc/meek.

[33] Named Data Networking Project. http://www.named-data.net.

[34] E. Nygren, R. K. Sitaraman, and J. Sun. The Akamai network: a platform for high-performance internet applications. *ACM SIGOPS OSR*, 44(3):2–19, 2010.

[35] A.-M. K. Pathan and R. Buyya. A taxonomy and survey of content delivery networks. http://www.cloudbus.org/reports/CDN-Taxonomy.pdf.

[36] D. Raychaudhuri, K. Nagaraja, and A. Venkataramani. Mobilityfirst: a robust and trustworthy mobility-centric architecture for the future internet. *ACM SIGMOBILE MCCR*, 16(3):2–13, 2012.

[37] P. Winter and S. Lindskog. How the Great Firewall of China Is Blocking Tor. In *FOCI*, 2012.

[38] E. Wustrow, S. Wolchok, I. Goldberg, and J. Halderman. Telex: Anticensorship in the Network Infrastructure. In *USENIX Security*, 2011.

[39] W. Zhou, A. Houmansadr, M. Caesar, and N. Borisov. SWEET: Serving the Web by Exploiting Email Tunnels. In *HotPETs*, 2013.