Draft of September 25, 2017

Chapter 1

Introduction

This is a thesis about Internet censorship. In it, I will expand on two threads of research that have occupied my attention for the past several years: better understanding how censors work, and fielding systems that circumvent their restrictions. These two threads fuel each other: better understanding censors enables us to build better circumvention systems that take into account their strengths and weaknesses; and the deployment of a circumvention system affords an opportunity to observe how censors themselves react to changing circumstances. If I am successful, the output of my research is useful models that describe not only how censors behave today but how they may evolve in the future, and tools for circumvention that are not only sound in theory but also effective in practice.

1.1 Scope

Censorship is an enormous topic. Even the addition of the "Internet" qualifier hardly reduces its scope, because almost everything that might be censored touches the Internet in some way. To deal with the subject in depth, it is necessary to limit the scope. My research is focused on an important special case of censorship, which I call the "border firewall" case. It is illustrated in Figure 1.1.

A *client* resides within a network that is entirely controlled by a *censor*. Within the

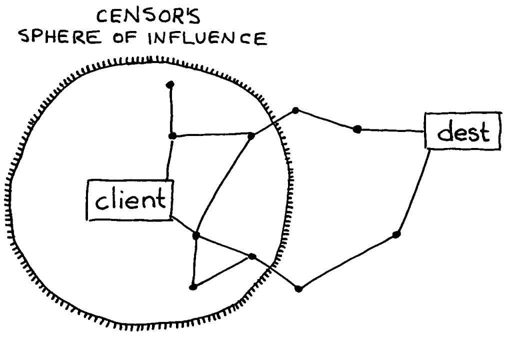

Figure 1.1: In the border firewall scenario, a client within a censor-controlled network wants to reach a destination that lies outside the censor's control.

censor's network, the censor may observe, modify, inject, or block any communication along any link. The censor, in particular, tries to prevent some subset of communication with the wider Internet, for instance by blocking certain keywords, network addresses, or protocols. The client's computer, however, is trustworthy and not controlled by the censor. The client's goal is to communicate with some *destination* that lies outside the censor's network, despite the censor's blocks: we call this activity *circumvention.* Circumvention requires somehow safely traversing a hostile network, eluding detection and blocking by the censor, in order to reach a destination. The censor does not control network links outside its own border; it may of course send messages to the outside world, but it cannot control them after they have traversed the border.

This abstract model is a good starting point, but the situation in practice is never so clear-cut. For example, the censor may be weaker than assumed: it may observe only the links at the border, not those wholly inside; it may not be able to fully inspect every packet that flows on its network; or there may be deficiencies or dysfunctions in its detection capabilities. Or the censor may be stronger: perhaps it, while not fully controlling outside networks, may influence their operators to discourage them from assisting in circumvention. The client may be limited, for technical or social reasons, in the software and hardware they can use. The destination may knowingly cooperate with the client's circumvention, or may not. There is no limit to the possible complications. Adjusting the basic model to reflect real-world actors' motivations and capabilities is the heart of threat modeling, one of the main topics of this thesis. Depending on the situation, we will add to the list of assumptions. In particular, what makes circumvention possible at all is the censor's motivation to block only *some* of the incoming and outgoing traffic, and allow the rest to pass—this assumption will be a major focus of the next chapter.

It is not hard to see how the border firewall model relates to censorship in practice. In a common case, the censor is a national government, and the borders of its controlled network correspond to the borders of a country. A government typically has the power to enforce laws and control network infrastructure to act within its own borders, but not outside. However the boundaries of censorship do not always correspond exactly to the border of a country. Almost since the study of Internet censorship began, it has been recognized that content restrictions may vary across geographic locations, even within the same country (Wright et al. [87] identified some possible causes). In some places a better model is not a single unified censorship regime, but rather many individual Internet service providers, each controlling its own network and acting as a mini-censor, perhaps coordinating with others about what to block and perhaps not. Another important case is that of a university or corporate network, in which the only outside network access is through a single gateway router, which tries to enforce a policy on what is acceptable and what is not. These smaller networks often differ from national- or ISP-level networks in interesting ways, for instance with regard to the amount of overblocking they are willing to tolerate, or the amount of computation they can afford to spend on each communication.

Here are some examples of forms of censorship that are in scope:

- blocking IP addresses
- blocking specific network protocols

- blocking DNS resolution for certain domains
- blocking keywords in URLs
- dissecting network layers ("deep packet inspection")
- statistical and probabilistic traffic classification
- active measures by censors to discover the use of circumvention

Other forms of censorship that are *not* in scope include:

- domain takedowns (that affect all clients globally)
- server-side blocking (servers refusing to serve certain clients)
- anything that takes place entirely within the censor's network and does not cross the border
- forum moderation and deletion of social media posts
- deletion-resistant publishing like the Eternity Service [5]—except insofar as access to such services may be blocked

Some have objected to the use of the generic term "Internet censorship" to refer to the narrow case of the border firewall. I am sensitive to this objection and acknowledge that far more topics could fit under the umbrella of Internet censorship. Nevertheless, for the purpose of this thesis, I will continue to use "Internet censorship" without further qualification to refer to the border firewall case.

Chapter 2

Principles of circumvention

In order to understand the challenges of circumvention, it helps to put yourself in the mindset of a censor. A censor has two high-level functions: detection and blocking. Detection is a classification problem: the censor prefers to permit some communications and deny others, and so it must have some procedure for deciding which communications fall in which category. Blocking follows detection. Once the censor detects some prohibited communication, it must take some action to stop the communication, such as terminating the connection at a network router. A censor must be able both to detect and to block. (Detection without blocking would be called not censorship, but surveillance.) The flip side of this statement is that a circumventor succeeds either by eluding detection, or, once detected, somehow resist the censor's blocking action. Research on circumvention has mostly dealt with the detection problem—a minority of research is on resisting blocking despite being detected.

A censor is, then, essentially a traffic classifier coupled with a blocking mechanism. Though the design space is large, and many complications are possible, at its heart it must decide, for each communication, whether to block or allow, and then effect blocks as appropriate. Any classifier is liable to make mistakes. When the censor fails to block something that it would have preferred to block, it is an error called a *false negative*; when the censor accidentally blocks something that it would have preferred to allow, it is a *false positive*. Forcing the censor to trade false negative for false positives is the core of all circumvention that is based on avoiding detection. Understanding the relative importance of misclassification errors to the censor—knowing what it prefers to allow and to block—is important for designing circumvention systems.

here be dragonſ

Detection ranges from almost trivial to very complicated.

detection can be trivial or complicated can be precomputed limits on scale may constrin what censors can do

"obfuscation" term is apt. not reflecting a mindset of security through obscurity; rather a recognition that it's about making the classification more difficult, and forcing the censor to trade false positives for false negatives.

The censor can block direct access to any destination, so circumvention typically uses, at minimum, some kind of indirect access, such as connecting through a proxy server.

Cite Pfitzmann + Hansen [68]: undetectability, unobservability, unblockability. Houmansadr?: entanglement. I prefer to think of it in terms of costs.

eavesdropper's dilemma [19] (as an example of having an empty sphere of visibility?) Ignoring the Great Firewall of China [16]: detection succeeds but not blocking. Flakiness of firewalls, etc. "blocking" include throttling, disruption more generally detection can include preprocessing

I find it helpful to break detection into two classes: detection by content and detection by address. ... The first is blocking by content; that is, by what you say. HTTP request keyword filtering and blocking based on deep packet inspection fall into this category. The second is blocking by address; that is, by whom you talk to. IP address blocking and DNS tampering fall into this category. The third is active probing, in which the censor imitates a client in order to discover proxy servers. Active probing is usually used as input for an address-blocking mechanism. Of these challenges, address blocking is probably the hardest, because it is efficient to implement in firewall hardware, and because network addresses are a scarcer resource than protocol variations.

Appendix 4.1 contains a summary of censorship circumvention systems and how they have changed over time in response to changing censorship threats.

This taxonomy of censorship techniques is not the only one possible. Philipp Winter divides it into three problems [83, § 1.1]: the bootstrapping problem; the endpoint blocking problem; and the traffic obfuscation problem. Khattak, Elahi, et al. [53] call these two tasks "fingerprinting" and "direct censorship"; Tschantz et al. [78] call them "detection" and "action."

2.1 Collateral damage

The cost of false positives is so important to circumvention that researchers have a specialized term for it: collateral damage. Collateral damage encompasses all the harm suffered by the censor through inadvertent, ancillary blocking done in the course of censorship. The term is a bit unfortunate, because it is easily misunderstood. If circumventors do things right, the potential "damage" is never realized, because the censor sees the cost as being too great. Circumventors try to make false positives so expensive that the censor has no choice but to allow false negatives; that is, to permit circumvention traffic.

collateral damage not a nice name means the same as "making the classification problem difficult" if you think of the censor as a classifier. false positive and false negative costs—circumventor's tactic is to bind FPs and FNs tightly together. underlies all circumvention according to the usual threat models (maybe not, in cases where censor can observe, but not influence) even look-like-something, stego transports ultimately depend on collateral damage (lengthy explanation and examples)

There are some forms of circumvention that do not rely on collateral damage; they are those in which the censor's sphere of influence is nil. That it, they rely on a channel that the censor is willing to block, but somehow actually unable to block. A hypothetical example might be a radio broadcast that the censor cannot jam because it lacks the necessary equipment. This is an example of a censor having an empty sphere of influence and a nonempty sphere of visibility: it can look, but not touch. A real-life example is Toosheh [77]... (also has receiver anonymity)

Don't need to be vague, saying that there is some communication the censor is unwilling to block. Make it concrete: this is what collateral damage the censor would have to incur to block this. If that collateral damage is large, then you win. Indistinguishability is a means toward increasing collateral damage. turn your assumptions into testable or quantifiable hypotheses don't say, "the censor cannot do X"; say, "in order to do X, the censor would have to..." make the threat models falsifiable: not just assumptions but hypotheses about how the world works (or will work)

2.2 Bridge distribution

Resistance to blocking by address; obfuscated protocol then prevents blocking by content.

- Kaleidoscope [76, 75]
- Mahdian [58]
- Proximax [61]
- rBridge [81]
- Salmon [25]
- Hyphae [56]
- Enemy at the Gateways [64]

In the usual threat models, though, the censor is assumed to be quite powerful, capable of dropping, replaying, and forging arbitrary packets, of . . . there is usually a concession to the censor needing to operate at line rate, or of needing to protect important communications (which is an argument about collateral damage), which provides the weakness that the circumvention system in question exploits. we already know that such a strong censor model is a fiction for national censors, for example the GFW acts like an "on-path" network monitor that can inject, but not drop, packets. the very strong threat model may be appropriate for e.g. whitelisting corporate or university censors

address blocking content blocking (could also separate out e.g. timing (and something else? check Khattak2016a)) The harder problem is address blocking: bridge distribution and rendezvous

The mass censors we know are weak if you are not being specifically targeted Pick a proxy server used by you and no one else Do any silly thing for obfuscation, it will work, because who cares There are true challenges in making it scale to large numbers of users and an adaptive adversary The cat-and-mouse game is not inevitable—don't think of it as "circumvention works until it gets popular, then it gets blocked" rather as "you get a free ride until you get popular, after that your thing has to actually work."

Generic rendezvous: BridgeDB and others

Traffic transformation look like nothing and look like something Psiphon anecdote about prepending HTTP to obfssh depending on physical aspects of networks Denali

infrastructure-based, decoy routing and domain fronting

Packet fragmentation tricks, etc. Cite brdgrd, [54]

pluggable transports

Tying questions of ethics to questions about censor behavior, motivation: [87] (also mentions "organisational requirements, administrative burden") [49] [17] Censors may come to conclusions different than what we expect (have a clue or not).

2.3 Early censorship and circumvention

Early censors (around the time of the late 1990s and early 2000s) would be considered weak by today's standards. They were mostly easy to circumvent by simple countermeasures, such as tweaking a protocol or using an alternative DNS server. But as censors become more capable, our models have to evolve to match. Indeed, my interest in threat modeling might be described as a sort of meta-modeling, learning about how threat models change over time and according to circumstances.

[16] [15] Thailand (1996, first?)

[45] [22] first report on DNS hijacking? Freedom House Freedom on the Net

anonymizer, dialectizer sites HTML rewriting proxies (BIFSO article predicting failure of censorship, leading to CGIProxy?)

changing dns servers

2.4 Open problems in censor modeling

Ongoing, longitudinal measurement of censorship remains a challenge. Studies tend to be limited to one geographical region and one period of time. Dedicated measurement platforms such as OONI [42] and ICLab [48] are starting to make a dent in this problem, by providing regular measurements from many locations worldwide. Even with these, there are challenges around getting probes into challenging locations and keeping them running.

Apart from a few reports of, for example, per annum spending on filtering hardware, not much is known about how much censorship costs to implement. In general, contemporary threat models tend to ignore resource limitations on the part of the censor.

Chapter 3

Measurement studies and measurement platforms

Analyzing Internet Censorship in Pakistan[1]

informing our threat models

censors' capabilities—presumed and actual e.g. ip blocking (reaction time?) active probing

Khattak and Elahi et al. [53] put it nicely with the terms "sphere of influence" and "sphere of visibility."

proxy-probe

Internet curfews (Gabon), limited time of shutdowns shows sensitivity to collateral damage.

commercial firewalls (Citizen Lab) and bespoke systems

Chapter 4

Studies of censors

This section surveys past measurement studies in order to draw specific and general conclusions about censor models. The objects of this survey are based on those in the evaluation study done by me and others in 2016 [78, §IV.A].

The main tool we have to build relevant threat models is the natural study of censors. The study of censors is complicated by difficulty of access: censors are not forthcoming about their methods. Researchers are obligated to treat censors as a black box, drawing inferences about their internal workings from their externally visible characteristics. The easiest thing to learn is the censor's *what*—the destinations that are blocked. Somewhat harder is the investigation into *where* and *how*, the specific technical mechanisms used to effect censorship and where they are deployed in the network. What we are really interested in, and what is most difficult to infer, is the *why*, or the motivations and goals that underlie a censorship apparatus. We posit that censors, far from being unitary entities of focused purpose, are rather complex organizations of people and machines, with conflicting purposes and economic rationales, subject to resource limitations. The *why* gets to the heart of why circumvention is even possible: a censoring firewall's duty is not merely to block, but to *discriminate* between what is blocked and what is allowed, in support of some other goal. Circumvention systems confuse this discrimination in order to sneak traffic through the firewall.

Past measurement studies have mostly been short-lived, one-off affairs, focusing deeply on one region of the world for at most a few months. Thus published knowledge about censors' capabilities consists mostly of a series of "spot checks" with blank areas between them. There have been a few designs proposed to do ongoing measurements of censorship, such as ConceptDoppler [18] in 2007 and CensMon [72] in 2011, but these have not lasted long in practice, and for the most part there is an unfortunate lack of longitudinal and cross-country measurements. Just as in circumvention, in censorship measurement a host of difficulties arise when running a scalable system for a long time, that do not arise when doing a one-time operation. The situation is thankfully becoming better, with the increasing data collection capabilities of measurement systems like OONI [42].

From the survey of measurement studies we may draw some general conclusions. Censors change over time, sometimes for unknown reasons, and not always in the direction of greater restrictions. Censorship conditions differ greatly across countries, not only in subject but in mechanism and motivation. The "Great Firewall" of China has long been the world's most sophisticated censor, but it is in many ways an outlier, and not representative of censors elsewhere. Most censors are capable of manipulating DNS responses, IP address blocking, and keyword filtering at some level.

A reasonable set of capabilities, therefore, that a contemporary censor may be assumed to have is:

- blocking of specific IP addresses and ports,
- control of default DNS servers,
- injection of false DNS responses,
- injection of TCP RSTs,
- throttling of connection,
- keyword filtering
- protocol identification, and
- temporary total shutdown of Internet connections

Not all censors will be able to do all of these. As the amount of traffic to be handled increases, in-path attacks such as throttling become relatively more expensive. Whether a particular censoring act even makes sense will depend on a local cost–benefit analysis. Some censors may be able to tolerate a brief total shutdown, while for others the importance of the Internet is too great for such a crude measure.

Past measurement studies have done a good job at determining the technical aspects of censorship, for example where in the network censorship routers are located. There is not so much known about the inner workings of censors. The anonymous paper on China's DNS censorship [7] probably comes closest to the kind of insight I am talking about, with its clever use of side channels to infer operational characteristics of censor boxes. For example, their research found that each DNS injection node runs a few hundred independent processes. This is indirect information, to be sure, but it hints at the level of resources the censor is able to bring to bear. I am interested in even deeper information, for example how censors make the decision on what to block, and what bureaucratic and other considerations might cause them to work less than optimally.

One of the earliest technical studies of censorship occurred not in some illiberal place, but in the German state of North Rhein-Westphalia. In 2003, Dornseif [23] tested ISPs' implementation of a controversial legal order to block two Nazi web sites. While there were many possible ways to implement the block, none were trivial to implement, nor free of overblocking side effects. The most popular implementation used *DNS tampering*, simply returning (or injecting) false responses to DNS requests for the domain names of the blocked sites. An in-depth survey of DNS tampering found a variety of implementations, some blocking more and some blocking less than required by the order.

Clayton [15] in 2006 studied a "hybrid" blocking system, called "CleanFeed" by the British ISP BT, that aimed for a better balance of costs and benefits: a "fast path" IP address and port matcher acted as a prefilter for the "slow path," a full HTTP proxy. The system, in use since 2004, was designed to block access to any of a secret list of pedophile web sites compiled by a third party. The author identifies ways to circumvent or attack such a system: use a proxy, use source routing to evade the blocking router, obfuscate requested URLs, use an alternate IP address or port, return false DNS results to put third parties on the "bad" list. They demonstrate that the two-level nature of the blocking system unintentionally makes it an oracle that can reveal the IP addresses of sites in the secret blocking list.

[21]

For a decade, the OpenNet Initiative produced reports on Internet filtering and surveillance in dozens of countries, until it ceased operation in 2014. For example, their 2005 report on Internet filtering in China [65] studied the problem from many perspectives, political, technical, and legal. They translated and interpreted Chinese laws relating to the Internet, which provide strong legal justifications for filtering. The laws regulate both Internet users and service providers, including cybercafes. They prohibit the transfer of information that is indecent, subversive, false, criminal, or that reveals state secrets. The OpenNet Initiative tested the extent of filtering of web sites, search engines, blogs, and email. They found a number of blocked web sites, some related to news and politics, and some on sensitive subjects such as Tibet and Taiwan. In some cases, entire sites (domains) were blocked; in others, only specific pages within a larger site were blocked. In a small number of cases, sites were accessible by IP address but not by domain name. There were cases of overblocking: apparently inadvertently blocked sites that simply shared an IP address or URL keyword with an intentionally blocked site. On seeing a prohibited keyword, the firewall blocked connections by injecting a TCP RST packet to tear down the connection, then injecting a zero-sized TCP window, which would prevent any communication with the same server for a short time. Using technical tricks, the authors inferred that Chinese search engines index blocked sites (perhaps having a special exemption from the general firewall policy), but do not return them in search results. The firewall blocks access searches for certain keywords on Google as well as the Google Cache—but the latter could be worked around by tweaking the format of the URL. Censorship of blogs comprised keyword blocking by domestic blogging services, and blocking of external domains such as blogspot.com. Email filtering is done by the email providers themselves, not by an independent network firewall. Email providers seem to implement their filtering rules independently and inconsistently: messages were blocked by some providers and not others.

In 2006, Clayton, Murdoch, and Watson [16] further studied the technical aspects of the Great Firewall of China. They relied on an observation that the firewall was symmetric, treating incoming and outgoing traffic equally. By sending web requests from outside the firewall to a web server inside, they could provoke the same blocking behavior that someone on the inside would see. They sent HTTP requests containing forbidden keywords (e.g., "falun") caused the firewall to inject RST packets towards both the client and server. Simply ignoring RST packets (on both ends) rendered the blocking mostly ineffective. The injected packets had inconsistent TTLs and other anomalies that enabled their identification. Rudimentary countermeasures such as splitting keywords across packets were also effective in avoiding blocking. The authors of this paper bring up an important point that would become a major theme of future censorship modeling: censors are forced to trade blocking effectiveness against performance. In order to cope with high load at a reasonable costs, censors may choose the architecture of a network monitor or intrusion detection system, one that can passively monitor and inject packets, but cannot delay or drop them.

A nearly contemporary study by Wolfgarten [85] reproduced many of the results of Clayton, Murdoch, and Watson. Using a rented server in China, the author found cases of DNS tampering, search engine filtering, and RST injection caused by keyword sniffing. Not much later, in 2007, Lowe, Winters, and Marcus [57] did detailed experiments on DNS tampering in China. They tested about 1,600 recursive DNS servers in China against a list of about 950 likely-censored domains. For about 400 domains, responses came back with bogus IP addresses, chosen from a set of about 20 distinct IP addresses. Eight of the bogus addresses were used more than the others: a whois lookup placed them in Australia, Canada, China, Hong Kong, and the U.S. By manipulating TTLs, the authors found that the false responses were injected by an intermediate router: the authentic response would be received as well, only later. A more comprehensive survey [7] of DNS tampering and injection occurred in 2014, giving remarkable insight into the internal structure of the censorship machines. DNS injection happens only at border routers. IP ID and TTL analysis show that each node is a cluster of several hundred processes that collectively inject censored responses. They found 174 bogus IP addresses, more than previously documented. They extracted a blacklist of about 15,000 keywords.

[86]

The Great Firewall, because of its unusual sophistication, has been an enduring object of study. Part of what makes it interesting is its many blocking modalities, both active and passive, proactive and reactive. The ConceptDoppler project of Crandall et al. [18] measured keyword filtering by the Great Firewall and showed how to discover new keywords automatically by latent semantic analysis, using the Chinese-language Wikipedia as a corpus. They found limited statefulness in the firewall: sending a naked HTTP request without a preceding SYN resulted in no blocking. In 2008 and 2009, Park and Crandall [66] further tested keyword filtering of HTTP responses. Injecting RST packets into responses is more difficult than doing the same to requests, because of the greater uncertainty in predicting TCP sequence numbers once a session is well underway. In fact, RST injection into responses was hit or miss, succeeding only 51% of the time, with some, apparently diurnal, variation. They also found inconsistencies in the statefulness of the firewall. Two of ten injection servers would react to a naked HTTP request; that it, one sent outside of an established TCP connection. The remaining eight of ten required an established TCP connection. Xu et al. [89] continued the theme of keyword filtering in 2011, with the goal of discovering where filters are located at the IP and AS levels. Most filtering is done at border networks (autonomous systems with at least one non-Chinese peer). In their measurements, the firewall was fully stateful: blocking was never triggered by an HTTP request outside an established TCP connection. Much filtering occurs at smaller regional providers, rather than on the network backbone.

Winter and Lindskog [84] did a formal investigation into active probing, a reported capability of the Great Firewall since around October 2011. They focused on the firewall's probing of Tor relays. Using private Tor relays in Singapore, Sweden, and Russia, they provoked active probes by simulating Tor connections, collecting 3295 firewall scans over 17 days. Over half the scan came from a single IP address in China; the remainder seemingly came from ISP pools. Active probing is initiated every 15 minutes and each burst lasts for about 10 minutes.

Sfakianakis et al. [72] built CensMon, a system for testing web censorship using PlanetLab nodes as distributed measurement points. They ran the system for for 14 days in 2011 across 33 countries, testing about 5,000 unique URLs. They found 193 blocked domain–country pairs, 176 of them in China. CensMon reports the mechanism of blocking. Across all nodes, it was 18.2% DNS tampering, 33.3% IP address blocking, and 48.5% HTTP keyword filtering. The system was not run on a continuing basis. Verkamp and Gupta [80] did a separate study in 11 countries, using a combination of PlanetLab nodes and the computers of volunteers. Censorship techniques vary across countries; for example, some show overt block pages and others do not. China was the only stateful censor of the 11.

PlanetLab is a system that was not originally designed for censorship measurement, that was later adapted for that purpose. Another recent example is RIPE Atlas, a globally distributed Internet measurement network consisting of physical probes hosted by volunteers, Atlas allows 4 types of measurements: ping, traceroute, DNS resolution, and X.509 certificate fetching. Anderson et al. [4] used Atlas to examine two case studies of censorship: Turkey's ban on social media sites in March 2014 and Russia's blocking of certain LiveJournal blogs in March 2014. In Turkey, they found at least six shifts in policy during two weeks of site blocking. They observed an escalation in blocking in Turkey: the authorities first poisoned DNS for twitter.com, then blocked the IP addresses of the Google public DNS servers, then finally blocked Twitter's IP addresses directly. In Russia, they found ten unique bogus IP addresses used to poison DNS.

Most research on censors has focused on the blocking of specific web sites and HTTP keywords. A few studies have looked at less discriminating forms of censorship: outright shutdowns and throttling without fully blocking. Dainotti et al. [20] reported on the total Internet shutdowns that took place in Egypt and Libya in the early months of 2011. They used multiple measurements to document the outages as they occurred: BGP data, a large network telescope, and active traceroutes. During outages, there was a drop in scanning traffic (mainly from the Conficker botnet) to their telescope. By comparing these different measurements, they showed that the shutdown in Libya was accomplished in more that one way, both by altering network routes and by firewalls dropping packets. Anderson [3] documented network throttling in Iran, which occurred over two major periods between 2011 and 2012. Throttling degrades network access without totally blocking it, and is harder to detect than blocking. The author argues that a hallmark of throttling is a decrease in network throughput without an accompanying increase in latency and packet loss, distinguishing throttling from ordinary network congestion. Academic institutions were affected by throttling, but less so than other networks. Aryan et al. [8] tested censorship in Iran during the two months before the June 2013 presidential election. They found multiple blocking methods: HTTP request keyword filtering, DNS tampering, and throttling. The most usual method was HTTP request filtering. DNS tampering (directing to a blackhole IP address) affected only three domains: facebook.com, youtube.com, and plus.google.com. SSH connections were throttled down to about 15% of the link capacity, while randomized protocols were throttled almost down to zero 60 seconds into a connection's lifetime. Throttling seemed to be achieved by dropping packets, thereby forcing TCP's usual recovery.

Khattak et al. [54] evaluated the Great Firewall from the perspective that it works like an intrusion detection system or network monitor, and applied existing technique for evading a monitor the the problem of circumvention. They looked particularly for ways to evade detection that are expensive for the censor to remedy. They found that the firewall is stateful, but only in the client-to-server direction. The firewall is vulnerable to a variety of TCP- and HTTP-based evasion techniques, such as overlapping fragments, TTL-limited packets, and URL encodings.

Nabi [63] investigated web censorship in Pakistan in 2013, using a publicly known list of banned web sites. They tested on 5 different networks in Pakistan. Over half of the sites on the list were blocked by DNS tampering; less than 2% were additionally blocked by HTTP filtering (an injected redirection before April 2013, or a static block page after that). They conducted a small survey to find the most commonly used circumvention methods in Pakistan. The most used method was public VPNs, at 45% of respondents.

Ensafi et al. [29] employed an intriguing technique to measure censorship from many locations in China—a "hybrid idle scan." The hybrid idle scan allows one to test TCP connectivity between two Internet hosts, without needing to control either one. They selected roughly uniformly geographically distributed sites in China from which to measure connectivity to Tor relays, Tor directory authorities, and the web servers of popular Chinese web sites. There were frequent failures of the firewall resulting in temporary connectivity, typically lasting in bursts of hours.

In 2015, Marczak et al. [59] investigated an innovation in the capabilities of the border routers of China, an attack tool dubbed the "Great Cannon." The cannon was responsible for denial-of-service attacks on Amazon CloudFront and GitHub. The unwitting participants in the attack were web browsers located outside of China, who began their attack when the cannon injected malicious JavaScript into certain HTTP responses originating in China. The new attack tool is noteworthy because it demonstrated previously unseen in-path behavior, such as packet dropping.

Not every censor is China, with its sophisticated homegrown firewall. A major aspect of censor modeling is that many censors use commercial firewall hardware. A case in point is the analysis by Chaabane et al. [13] of 600 GB of leaked logs from Blue Coat proxies used for censorship in Syria. The logs cover 9 days in July and August 2011, and contain an entry for every HTTP request. The authors of the study found evidence of IP address blocking, domain name blocking, and HTTP request keyword blocking, and also of users circumventing censorship by downloading circumvention software or using the Google cache. All subdomains of .il, the top-level domain for Israel, were blocked, as were many IP address ranges in Israel. Blocked URL keywords included "proxy", "hotspotshield", "israel", and "ultrasurf" (resulting in collateral damage to the Google Toolbar and Facebook Like button because they have "proxy" in HTTP requests). Tor was only lightly censored—only one of several proxies blocked it, and only sporadically.

4.1 Summary of circumvention systems

Many circumvention systems have been proposed or deployed. My survey with Tschantz, Afroz, and Paxson [78] covered 54 systems; a later one by Khattak, Elahi, et al. [53] covered 73. The systems mentioned in this section are not exhaustive but are chosen to be representative.

Against content blocking, circumvention systems generally take one of two strategies. The first is steganography, trying to blend in with some other protocol that the censor does not already block. The second is polymorphism, trying to look unlike anything the censor already blocks. Which one is more appropriate depends on the censor model. Against a censor that whitelists a small number of protocols and prohibits everything else, steganography is appropriate. Against a censor that blacklists a small number of protocols or keywords, polymorphism is appropriate. (The common understanding is that real-world censors tend to be of the blacklisting type, because whitelisting causes too much inherent collateral damage—it is too hard to enumerate all the protocols users might want to use. The exception is in exceptionally constrained networks such as that of Cuba, that do not derive as much benefit from Internet connectivity anyway, and so can afford the collateral damage.)

FTE [27] (for "format-transforming encryption") is a quintessential example of a steganographic protocol. Given a specification of a regular expression, FTE transforms traffic to match it. The purpose is to force false-negative misclassification by firewalls. StegoTorus [82] uses custom encoders to make traffic resemble common HTTP file types, such as PDF, JavaScript, and Flash. FreeWave [47] modulates a data stream into an acoustic signal and transmits it over VoIP.

The history of the polymorphic, randomized protocols known as obfs2 [50], obfs3 [51], and obfs4 [6] is interesting because it tells a story of circumventors changing behavior in the face of changing censor models. All of these protocols aim to encode traffic as a uniformly random sequence of bytes, leaving no plaintext features for a censor to detect. The obfs2 protocol used a fairly naive handshake protocol that appeared random only to a first approximation. It would have bypassed the keyword- or pattern-based censors of its era, but it was detectable passively, using a custom detector. obfs3 improved on obfs2 by adding a clever Diffie–Hellman key exchange, specially modified to also appear random to a censor. obfs3 was not trivially detectable passively, but could be attacked by an active man in the middle, and was vulnerable to active probing. obfs4 added an out-of-band secret that foils both man-in-the-middle and active probing attacks.

"Decoy routing" systems put proxies at the middle of network paths. A special cooperating router lies between the client and the apparent destination of a TCP stream. The router looks for a special cryptographic "tag" that is undetectable to the censor. On finding a tag, the router begins to redirect the client's traffic away from its declared destination and towards a censored destination instead. There are several decoy routing proposals, each with advantages and disadvantages; those that began the line of research are called Curveball [52], Telex [88], and Cirripede [46].

Chapter 5

Empirically testing real-world censors

In 2015 I helped study the phenomenon of “active probing” by the Great Firewall to discover hidden proxy servers. In active probing, the censor pretends to be a legitimate client of the proxy server: it connects to suspected servers to check whether they speak a proxy protocol. If they do, then they are blocked. Active probing makes good sense for the censor: it has high precision (low risk of collateral damage), and is efficient because it can be run as a batch job apart from a firewall’s real-time responsibilities. The Great Firewall can dynamically active-probe and block the servers of a number of common circumvention protocols, such as Tor, obfs2, and obfs3, within only seconds or minutes of a connection by a legitimate client. The need to resist active probing has informed the design of recent circumvention systems, including meek.

My primary contribution to the active probing project was the analysis of server logs to uncover the history of about two and a half years of active probing. My work revealed the wide distribution of active probing source addresses (there were over 14,000 of them). It also discovered previously undocumented types of probes, for the protocol used by VPN Gate and for a simple form of domain-fronted proxy. I helped analyze the network “fingerprints” of active probes and how they might be distinguished from connections by legitimate clients.

The work on active probing appeared in the 2015 research paper “Examining How the Great Firewall Discovers Hidden Circumvention Servers” [28], which I coauthored with Roya Ensafi, Philipp Winter, Nick Feamster, Nicholas Weaver, Vern Paxson.

I am interested in understanding censors at a deeper level. To that end, I am working on a project to measure how long censors take to react to sudden changes in circumvention. So far, our technique has been to monitor the reachability of newly added Tor Browser bridges, to see how long after they are introduced they get blocked. Portions of this work have already appeared in the 2016 research paper “Censors’ Delay in Blocking Circumvention Proxies” [41], which I coauthored with Lynn Tsai. We discovered some interesting, previously undocumented behaviors of the Great Firewall of China. While the firewall, through active probing, is able to detect some bridges dynamically within seconds or minutes, it lags in detecting Tor Browser’s newly added bridges, taking days or weeks to block them. It seems that bridges are first blocked only at certain times of day, perhaps reflecting an automated batch operation.

I am now continuing to work on this project along with Lynn Tsai and Qi Zhong. We plan to run targeted experiments to find out more about how censors extract bridge addresses from public information, for example, by adding bridges with different attributes and seeing whether they are blocked differently. Our first experiment used measurement sites only in China and Iran, but we hope to expand to many more countries by collaborating with measurement platforms such as OONI [42] and ICLab [48]. We hope to solicit other kinds of censor delays from other circumvention projects, in order to build a more all-encompassing picture of censors' priorities with respect to circumvention.

Chapter 6

Domain fronting

My most influential contribution to the world of circumvention is my research on domain fronting. While the basic idea is not mine, the research I led and the code I wrote helped domain fronting become the ubiquitous tool it is today.

Domain fronting assumes a rather strong censor model, essentially equivalent to the state of the art of national censors at the time of its popularization. That is, a censor that can block IP addresses and domain names, that can filter plaintext HTTP, can fingerprint protocol implementations. The main censor capabilities not provided for are probabilistic classification by traffic flow characteristics, and high-collateral-damage blocking of HTTPS on important web servers. What I find most intellectually compelling about domain fronting research is that is finally begins to transcend the "cat-and-mouse" paradigm that has plagued thinking around circumvention, and to put blocking resistance on a scientific basis. By this I mean that one can state assumptions, and consequences that hold as long as the assumptions are true. For example, we do not make claims such as "domain fronting is unblockable"; rather, we may state hypotheses and consequents: "if fronting through a domain with sufficient collateral damage, such that the censor is unwilling to block it, and if the censor does not find some side channel that distinguishes fronted from non-fronted traffic, then the communication will be unblocked." This kind of thinking, that of weighing censors' *costs* and *capabilities*, underlies my thinking about threat modeling.

Like flash proxy, domain fronting is primarily targeted at the problem of address blocking (though it is effective against content blocking and active probing as well). The core idea is the use of different domain names at different layers of communication. The "outside" layers, those visible to the censor, contain an innocuous "front" domain name, ideally one that is hard to block because of the value of the services behind it. The "inside" layer,

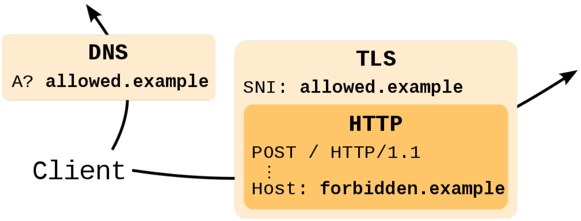

Figure 6.1: Domain fronting uses different names at different network layers.

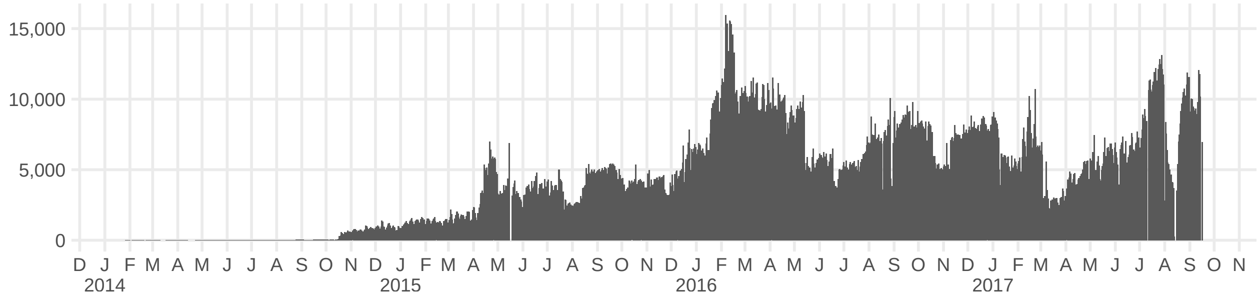

Figure 6.2: Estimated mean number of concurrent users of the meek pluggable transport.

invisible to the censor under encryption, contains the true, presumably censored, destination. An intermediate server, whose name is the front domain name, removes the outer layer of encryption and forwards the information to the covert destination. There are a number of important services that support domain fronting, mainly cloud providers and content delivery networks. On top of this basic machinery, it is relatively easy to build a general-purpose covert bidirectional communications channel, one that can even be made reasonably efficient.

I wrote and continue to maintain the code of meek, a circumvention transport for Tor based on domain fronting. It first appeared in Tor Browser in October 2014, and continues operation to the present. My code has been forked and incorporated by other circumvention projects, notably including Psiphon and Lantern, with whom I continue to collaborate. Today, meek is Tor's second-most-used transport, carrying around 10 terabytes of user traffic each month.

Domain fronting appeared in the 2015 research paper "Blocking-resistant communication through domain fronting" [39], which I coauthored with Chang Lan, Rod Hynes, Percy Wegmann, and Vern Paxson.

6.1 An unvarnished history of meek deployment

Fielding a circumvention and keeping it running is full of unexpected challenges. At the time of the publication of the domain fronting paper [39] in 2015, meek had been deployed only a year and a half. Here I will recount the entire history of the deployment project, from inception to the present, a period of over three years. I have been the main developer and project leader of meek over its entire existence. I hope to share the benefit of my experience by commentating the history with surprises and lessons learned. Figure 6.2 shows the estimated concurrent number of users of meek over its entire existence. The counts come from Tor Metrics [55].

The prehistory of meek begins in 2013 with flash proxy. Flash proxy clients need a secure way to register their address to a central facilitator, in order that flash proxies can connect back to them. Initially we had only two means of registration: flashproxy-reg-http, sending client registrations directly over HTTP; and flashproxy-reg-email, sending client registrations to a special email address. We knew that flashproxy-reg-http was easily blockable; flashproxy-reg-email had good blocking resistance but was somewhat slow and complicated, requiring a server to poll for new messages. At some point, Jacob Appelbaum showed me an example of using domain fronting—though we didn't have a name for it then—to access a simple HTTP-rewriting proxy on App Engine. I eventually realized that the same trick would work for flash proxy rendezvous. I proposed a design [10] in May 2013 and within a month Arlo Breault had written flashproxy-reg-appspot, which worked just like flashproxy-reg-http, but fronted through www.google.com rather than contacting the registration server directly. The fronting-based registration became flash proxy's preferred method, being faster and simpler than the email-based one.

The development into a full-fledged bidirectional transport seems slow, in retrospect. All the pieces were there; it was only a matter of putting them together. I did not appreciate the potential of domain fronting when I saw it for the first time. Even after the introduction of flashproxy-reg-appspot, months passed before the beginning of meek. The whole idea behind flash proxy registration was that the registration channel could be of low quality—unidirectional, low-bandwidth, and high-latency—because it was only used to bootstrap into a more capable channel (WebSocket). Email fits well into this model: not good for a general-purpose channel, just good enough for rendezvous. The fronting-based HTTP channel, however, was much more capable, bidirectional with reasonably high performance. Rather than handing off the client to a flash proxy, it should be possible to carry all the client's traffic through the same domain-fronted channel. It was during this time that I first became aware of GoAgent through the "Collateral Freedom" report of Robinson et al. [71]. According to the report, GoAgent, which used a less secure form of domain fronting than what meek would have, was the most used circumvention tool among a group of users in China. I read the source code of GoAgent in October 2013 and wrote ideas about writing a similar pluggable transport [31] which would become meek.

I lost time in premature optimization of meek's network performance. I was thinking about the request–response nature of HTTP, and how requests and responses could conceivably arrive out of order (even if reordering was unlikely to occur in practice, because of the keepalive connections and HTTP pipelining). I made several attempts at a TCP-like reliability and sequencing layer, none of which were satisfactory. I wrote a simplified experimental prototype called "meeker," which simply prepended an HTTP header before the client and server streams, but meeker only worked for direct connections, not through an HTTP-aware intermediary like App Engine. When I explained these difficulties to George Kadianakis in December 2013, he advised me to forget the complexity and implement the simplest thing that could work, which was good advice. I started working on a version that strictly serialized request–response pairs, which architecture meek still uses today.

6.1.1 2014

According to the Git revision history, I started working on the source code of meek proper on January 26, 2014. I made the first public announcement on January 31, 2014, in a post to the tor-dev mailing list titled "A simple HTTP transport and big ideas" [30]. (If the development time seems short, it's only because months of prototypes and false starts.) In the post, I linked to the source code, described the protocol, and explained how to try it, using an App Engine instance I had set up shortly before. At this time there was no web browser TLS camouflage, and only App Engine was supported. I was not yet using the term "domain fronting." The "big ideas" of the title were as follows: we could run one big public bridge rather than relying on multiple smaller bridges as other transports did; a web server with a PHP "reflector" script could do the same forwarding as a CDN, providing a diversity of access points even without domain fronting; we could combine meek with authentication and serve a 404 to unauthenticated users; and Cloudflare and other CDNs are alternatives to App Engine. We did end up running a public bridge for public benefit (and worrying over how to pay for it), and deploying on platforms other than App Engine (with Tor we never used Cloudflare specifically, but did others). Arlo Breault would write a PHP reflector, though there was never a repository of public meek reflectors as there were for other types of Tor bridge. Combining meek with authentication never happened; it was never needed for our public domain-fronted instances because active probing doesn't help the censor in those cases anyway.

During the spring 2014 semester (January–May) I was enrolled in Vern Paxson's Internet/Network Security course along with fellow student Chang Lan. We made the development and security evaluation of meek our course project. During this time we built browser TLS camouflage extensions, tested and polished the code, and ran performance tests. Our final report, "Blocking-resistant communication through high-value web services," was the kernel of our later paper on domain fronting.

In March 2014, I met some developers of Lantern at a one-day hackathon sponsored by OpenITP [11]. Lantern developer Percy Wegmann and I realized that the meek code I had been working on could act as a glue layer between Tor and the HTTP proxy exposed by Lantern, in effect allowing you to use Lantern as a pluggable transport for Tor. We worked out a prototype and wrote a summary of the process [33]. Even though our specific application that day did not use domain fronting, the early contact with other circumvention developers was valuable.

June 2014 brought a surprise: the Great Firewall of China blocked all Google services [43, 2]. It would be hubris to think that it was in response to the nascent deployment of meek on App Engine; a more likely cause was Google's decision to start using HTTPS for web searches, which would foil URL keyword filtering. Nevertheless, the blocking cast doubt on the feasibility of domain fronting: I had believed that blocking all of Google would be too costly in terms of collateral damage to be sustained for long by any censor, even the Great Firewall, and that belief was wrong. At least, we now needed fronts other than Google in order to have any claim of effective circumvention in China. For that reason, I set up additional backends: Amazon CloudFront and Microsoft Azure. When meek made its debut in Tor Browser, it would offer three modes: meek-google, meek-amazon, and meek azure.

Google sponsored a summit of circumvention researchers in June 2014. I presented domain fronting there. (By this time I had started using the term "domain fronting," realizing that what I had been working on needed a specific name. I tried to separate the idea "domain fronting" from the implementation "meek," but the terms have sometimes gotten confused in discourse.) Developers from Lantern and Psiphon where there—I was pleased to learn that Psiphon had already implemented and deployed domain fronting, after reading my mailing list posts. The meeting started a fruitful collaboration: Percy Wegmann from Lantern and Rod Hynes from Psiphon would later be among my coauthors on the paper on domain fronting [39].

Chang, Vern, and I submitted a paper on domain fronting to the Network and Distributed System Security Symposium (NDSS) in August 2014, whence it was rejected.

The first public release of Tor Browser that had a built-in easy-to-use meek client was version 4.0-alpha-1 on August 12, 2014 [14]. This was an alpha release, used by fewer users than the stable release. I made a blog post explaining how to use it a few days later [32]. The release and blog post had a positive effect on the number of users, however the absolute numbers are uncertain, because of a configuration error I had made on the meek bridge. I was running the meek bridge and the flash proxy bridge on the same instance of Tor; and because of how Tor's statistics are aggregated, the counts were spuriously correlated [35]. I switched the meek bridge to a separate instance of Tor on September 15; numbers after that date are more trustworthy. In any case, the usage before this first release was tiny: the App Engine bill ($0.12/GB, with one GB free each day) was less than $1.00 per month for the first seven months of 2014 [62, § Costs]. In August, the cost started to be nonzero every day, and would continue to rise from there.

Tor Browser 4.0 [67] was released on October 15, 2014. It was the first stable (not alpha) release to have meek, and it had an immediate effect on the number of users: the estimate jumped from 50 to 500 within a week. (The increase was partially conflated with a failure of the meek-amazon bridge to publish statistics before that date, but the other bridge, servicing meek-google and meek-azure, individually showed the same increase.) It was a lesson in user behavior: although there had been a working implementation in the alpha release for two months already, evidently a large number of users did not know of it or chose not to try it. At that time, the other transports available were obfs3, FTE, ScrambleSuit, and flash proxy.

6.1.2 2015

Through the first part of 2015, the estimated number of simultaneous users continued to grow, reaching about 2,000, as we fixed bugs and Tor Browser had further releases.

We submitted a revised version of the domain fronting [39], now with contributions from Psiphon and Lantern, to the Privacy Enhancing Technologies Symposium, where it was accepted and appeared on June 30 at the symposium.

The increasing use of domain fronting by various circumvention tools begain to attract more attention. A March 2015 article by Eva Dou and Alistair Barr in the Wall Street Journal [24] described domain fronting and "collateral freedom" in general, depicting cloud service providers as being caught in the crossfire between censors and circumventors. The journalists had contacted me but I declined to be interviewed. The CEO of CloudFlare, through whose service Lantern had been fronting, said that recently they had altered their systems to prevent domain fronting by enforcing a match between SNI and Host header [69]. GreatFire, an anticensorship organization that had also been mentioned, shortly thereafter experienced a new type of denial-of-service attack [73], caused by a Chinese network attack system later called the "Great Cannon" [60]. They blamed the attack on the attention brought by the news article.

Since initial deployment, the Azure backend had been slower, with fewer users, than the other two options, App Engine and CloudFront. For months I had chalked it up to limitations of the platform. In April 2015, though, I found the real source of the problem: the code I had written to run on Azure, the code that receives domain-fronted HTTP requests and forwards them to the meek bridge, was not reusing TCP connections. For every outgoing request, the Azure code was doing a fresh TCP and TLS handshake—causing a bottleneck at the CPU of the bridge, coping with all the incoming TLS. When I fixed the Azure code to reuse connections [9], the number of users (overall, not only for Azure) had a sudden jump, reaching 6,000 in less than a week. Evidently, we had been leaving users on the table by having one of the backends not run as fast as possible.

The deployment of domain fronting was being partly supported by a $500/month grant from Google. Already the February 2015, the monthly cost for App Engine alone began to exceed that amount [62, § Costs]. In an effort to control costs, in May 2015 we began to rate-limit the App Engine and CloudFront bridges, deliberately slowing the service so that fewer would use it. Until October 2015, the Azure bridge was on a research grant provided by Microsoft, so we allowed it to run as fast as possible, but when the grant expired, we rate-limited the Azure bridge as well. The rate-limiting explains the relative flatness of the user graph from May to the end of 2015.

Google changed the terms of service governing App Engine in 2015, adding a paragraph that seemed to prohibit running a proxy service [44]:

> *Networking.* Customer will not, and will not allow third parties under its control to: (i) use the Services to provide a service, Application, or functionality of network transport or transmission (including, but not limited to, IP transit, virtual private networks, or content delivery networks); or (ii) sell bandwidth from the Services.

This was an uncomfortable time: we seemed to have the support of Google, but the terms of service said otherwise. I contacted Google and asked for clarification or guidance, in the meantime leaving meek-google running; however I never got an answer to my questions. The point became moot a year later, when Google shut down our App Engine project, for another reason altogether.

By this time we had not received any reports of any type of blocking of domain fronting. We did, however, suffer a few accidental outages (which look just like blocking, from a user's point of view). Between July 20 and August 14, an account transition error left the Azure configuration broken [34]. I set up another configuration on Azure and published instructions on how to use it, but it would not be available to the majority of users until the next release of Tor Browser, which happened on August 11. Between September 30 and October 9, the CloudFront-fronted bridge was effectively down because of an expired TLS certificate. When it rebooted on October 9, an adminstrative oversight caused its Tor relay identity fingerprint changed—meaning that clients expecting the former fingerprint would refuse to connect to it [38]. The situation was not fully resolved until November 4 with the next release of Tor Browser: cascading failured led to over a month of downtime.

One of the benefits of building a circumvention system for Tor is the easy integration with Tor Metrics—the source of the user number estimates in this section. Since the beginning of meek's deployment, we had known about a problem with the way it integrates with Tor Metrics' data collection. Tor pluggable transports geolocate the client's IP address in order to aggregate statistics by country. But when a meek bridge receives a connection, the "client IP address" it sees is not that of the true client, but rather is some cloud server, the intermediary through which the domain-fronted traffic passes. So the total counts were fine, but the per-country counts were meaningless. For example, because App Engine's servers were located in the U.S., every meek-google connection was being counted in the U.S. bucket.

By the end of 2015, meek users were a large enough fraction (about 20%) of all bridge users, that they were really starting to skew the overall per-country counts. I wrote a patch to have the client's true IP address forwarded through the network intermediary in a special HTTP header, which fixed the per-country counts from then on.

6.1.3 2016

In mid-January 2016 the Tor Project asked me to raise the rate limits on the meek bridges, in anticipation of rumored attempts to block Tor in Egypt. (The blocking attempts were in turn rumored to be caused by Facebook's integration of Tor into their mobile application.) I had the bridge operators raise the rate limits from approximately 1 MB/s to 3 MB/s. The effect of the relaxed rate limits was immediate: the count shot up as high 15,000 simultaneous users, briefly becoming Tor's most-used pluggable transport, before settling in around 10,000.

The first action that may have been a deliberate attempt to block domain fronting came on January 29, 2016, when the Great Firewall of China blocked one of the edge servers of the Azure CDN. The blocking was by IP address, a severe method: not only the domain name we were using for domain fronting, but also thousands of other names, became inaccessible. The block lasted about four days. On February 2, the server changed its IP address, incrementing the final octet from .200 to .201, causing it to become unblocked. I am aware of no similar incidents before or since.

The next surprise was on May 13, 2016. meek's App Engine backend stopped working and I got a notice saying:

> We've recently detected some activity on your Google Cloud Platform/API Project ID meek-reflect that appears to violate our Terms of Service. Please take a moment to review the Google Cloud Platform Terms of Service or the applicable Terms of Service for the specific Google API you are using.
>
> Your project is being suspended for committing a general terms of service violation.
>
> We will delete your project unless you correct the violation by filling in the appeals form available on the project page of Developers Console to get in touch with our team so that we can provide you with more details.

My first thought was that it had to do with the changes to the terms of service that had happened the previous year—but the true cause was unexpected. I tried repeatedly to contact Google and learn the nature of the "general" violation, but was stonewalled. None of my inquiries received so much as an acknowlegement. It as not until June 18 that I got some insight as to what happened, through an unofficial channel. Some botnet had apparently been misusing meek for command and control purposes; and its operators hadn't even bothered to set up their own App Engine project. They were using the service that we had been operating for the public. Although we may have been able to reinstate the meek-google service, seeing as the suspension was the result of someone else's botnet, with the already uncertain standing with regard to the terms of service I didn't have the heart to pursue it. meek-google remained off and users migrated to meek-amazon or meek-azure. It turned out, later, that it had been no common botnet misusing meek-google, but an organized political hacker group, known as Cozy Bear or APT29. Matthew Dunwoody presented observations to that effect in a FireEye blog post [26] in March 2017. He and Nick Carr had presented those findings at DerbyCon in September 2016, but I was not aware of them until the blog post. Malware would install a backdoor that operated over a Tor onion service, and used meek for camouflage.

TLS fingerprinting Cyberoam May 2016 https://groups.google.com/d/topic/traffic-obf/BpFSCVgi5rs https://lists.torproject.org/pipermail/tor-talk/2016-May/040898.html FortiGuard July 2016 https://groups.google.com/d/topic/traffic-obf/fwAN-WWz2Bk Kazakhstan ¡ Dec 2016 https://trac.torproject.org/projects/tor/ticket/20348#comment:142

Brazil

Cert reload by Yawning Angel, Let's Encrypt support based on a patch by George Tankersley.

GAEuploader

Funding sources

Chapter 7

Building circumvention systems

Over the past five years I have been involved in the development of four noteworthy circumvention designs:

- Flash proxy [37], based on temporary proxies running in web browsers.
- OSS [40], using third-party web scanning services.
- Domain fronting [39], using popular web services for cover.
- Snowflake [74, 36] (in progress), based on peer-to-peer proxies in web browsers; flash proxy redux.

These have evolved according to the needs of the time and my growing understanding of how censorship should be modeled.

My main interest is resistance to address blocking, which I regard as more difficult to achieve than resistance to content blocking. The first two systems, flash proxy and OSS, made no special effort to avoid their content being blocked, leaving content obfuscation to be done by another layer. My later designs have taken the threats of content blocking and active probing more integrally into account.

7.1 Flash proxy, a circumvention system

I began working on censorship circumvention with flash proxy in 2011. Flash proxy is targeted at the difficult problem of proxy address blocking: it is designed against a censor model in which the censor can block any IP address it chooses, but only on a relatively slow timeline of several hours.

Flash proxy works by running tiny JavaScript proxies in ordinary users' web browsers. The mini-proxies serve as temporary stepping stones to a full-fledged proxy, such as a Tor relay. The idea is that the flash proxies are too numerous, diverse, and quickly changing to block effectively. A censored user may use a particular proxy for only seconds or minutes before switching to another. If the censor manages to block the IP address of one proxy, there is little harm, because many other temporary proxies are ready to take its place.

The flash proxy system was designed under interesting constraints imposed by being partly implemented in JavaScript in the browser. The proxies sent and received data using the WebSocket protocol, which allows for socket-like persistent TCP connections in browsers, but with a catch: the browser can only make outgoing connections, not receive incoming ones as a traditional proxy would. The censored client must somehow inform the system of its own public address, and then the proxy connects *back* to the client. This architectural constraint was probably the biggest impediment to the usability of flash proxy, because it required users to configure their local router to permit incoming connections. (Removing this impediment is the main reason for the development of Snowflake, described later.) Flash proxy does not itself try to obfuscate patterns in the underlying traffic; it only provides address diversity.

For the initial "rendezvous" step in which a client advertises its address and a request for proxy service, flash proxy uses a neat idea: a low-capacity, but highly covert channel bootstraps the high-capacity, general-purpose WebSocket channel. For example, we implemented an automated email-based rendezvous, in which the client would send its address in an encrypted email to a special address. While it is difficult to build a useful low-latency bidirectional channel on top of email, email is difficult to block and it is only needed once, at the beginning of a session. We later replaced the email-based rendezvous with one based on domain fronting, which would later inspire meek, described below.

I was the leader of the flash proxy project and the main developer of its code. Flash proxy was among the first circumvention systems built for Tor—only obfs2 is older. It was first deployed in Tor Browser in January 2013, and was later retired in January 2016 after it ceased to see appreciable use. Its spirit lives on in Snowflake, now under development.

Flash proxy appeared in the 2012 research paper "Evading Censorship with Browser-Based Proxies" [37], which I coauthored with Nate Hardison, Jonathan Ellithorpe, Emily Stark, Roger Dingledine, Phil Porras, and Dan Boneh.

7.2 OSS, a circumvention prototype

OSS, for "online scanning service," is a design for circumvention based on the use of third-party web services that issue HTTP requests to user-specified destinations, such as an online translation service. OSS is designed against the model of a censor that is unwilling to block useful web services that are used for circumvention, because of the useful service they provide.

In OSS, the client sends messages to a censored destination by bouncing them through a third-party scanning service. The key idea is a deliberate conflation of address and content. The client asks the scanning service to scan a long URL that is constructed to encode both the destination host and a data payload. The destination receives the HTTP request and decodes its payload. The destination sends data downstream by abusing HTTP redirection, instructing the scanning service to send another HTTP request back to the client, with a different payload. The resistance to blocking of the OSS system hinges on the abundance of online scanning services that exist.

OSS was never deployed to users. I judged its overhead and potential to annoy webmasters to be too great to be practical. The core idea, however, did see use as a rendezvous method for flash proxy. In this method, a helper program would encode the client's IP address into a URL. The user would then copy and paste the URL into any online scanning service, which would then forward the information to the flash proxy system. In fact, this URL encoding was used internally by the domain fronting–based rendezvous as well, using a URL as a convenient vehicle for data transfer.

OSS appeared in the 2013 research paper "OSS: Using Online Scanning Services for Censorship Circumvention" [40], which I coauthored with Gabi Nakibly and Dan Boneh.

7.3 Snowflake, a circumvention system

I am working on a new circumvention system, a transport for Tor called Snowflake. Snowflake is the successor to flash proxy. It keeps the basic idea of in-browser proxies while fixing the usability problems that hampered the adoption of flash proxy. My main collaborators in this project are Serene Han and Arlo Breault.

The key difference between flash proxy and Snowflake is the basic communications protocol between client and browser proxy. Flash proxy used the TCP-based WebSocket protocol, which required users to configure their personal firewall to allow incoming connections. Snowflake instead uses WebRTC, a UDP-based protocol that enables peer-to-peer connections without manual configuration. The most similar existing system is uProxy [79], which in one of its operating modes uses WebRTC to connect through a friend's computer. Snowflake differs because it does not require prior coordination with a friend before connecting. Instead, it pulls its proxies from a pool of web users who are running the Snowflake code. Beyond the changed protocol, we hope to build in performance and efficiency improvements.

Snowflake will afford interesting research opportunities. One, of course, is the design of the system itself—no circumvention system of its nature has previously been deployed at a large scale. Another opportunity is observing how censors react to a new challenge.

Most of the available documentation on Snowflake is linked from the project's wiki page [74]. Mia Gil Epner and I wrote a preprint on the fingerprinting hazards of WebRTC [36].

Chapter 8

How circumvention technologies are evaluated

Evaluating the quality of circumvention systems is tricky, whether they are only proposed or actually deployed. The problem of evaluation is directly tied to threat modeling. Circumvention is judged according to how well it works under a given model; the evaluation is therefore meaningful only as far as the threat model reflects reality. Without grounding in reality, researchers risk running an imaginary arms race that evolves independently of the real one.

This kind of work is rather different than the direct evaluations of circumvention tools that have happened before, for example those done by the Berkman Center [70] and Freedom House [12] in 2011. Rather than testing tools against censors, we evaluated how closely calibrated designers' own models were to models derived from actual observations of censors.

This research was partly born out of frustration with some typical assumptions made in academic research on circumvention, which we felt placed undue emphasis on steganography and obfuscation of traffic streams, while not paying enough attention to the perhaps more important problems of bridge distribution and rendezvous. Indeed, in our survey of over 50 circumvention tools, we found that academic designs tended to be concerned with detection in the steady state after a connection is established, while actually deployed systems cared more about how the connection is established initially. We wanted to help bridge the gap by laying out a research agenda to align the incentives of researchers with those of circumventors. This work was built on extensive surveys of circumvention tools, measurement studies, and known censorship events against Tor.

This work on evaluation appeared in the 2016 research paper "Towards Grounding Censorship Circumvention in Empiricism" [78], which I coauthored with Michael Carl Tschantz, Sadia Afroz, and Vern Paxson.

Do they check the right things?

what's used and what's not used

8.1 Flash proxy

8.2 Domain fronting and meek

8.3 Snowflake: flash proxy revisited

WebRTC fingerprinting

Bibliography

[1] Giuseppe Aceto, Alessio Botta, Antonio Pescapè, M. Faheem Awan, Tahir Ahmad, and Saad Qaisar. "Analyzing Internet Censorship in Pakistan". In: *Research and Technologies for Society and Industry*. IEEE, 2016. http://wpage.unina.it/giuseppe.aceto/pub/aceto2016analyzing.pdf (cit. on p. 8).

[2] Percy Alpha. *Google disrupted prior to Tiananmen Anniversary; Mirror sites enable uncensored access to information*. June 2014. https://en.greatfire.org/blog/2014/jun/google-disrupted-prior-tiananmen-anniversary-mirror-sites-enable-uncensored-access (cit. on p. 21).

[3] Collin Anderson. *Dimming the Internet: Detecting Throttling as a Mechanism of Censorship in Iran*. Tech. rep. University of Pennsylvania, 2013. https://arxiv.org/pdf/1306.4361v1.pdf (cit. on p. 13).

[4] Collin Anderson, Philipp Winter, and Roya. "Global Network Interference Detection over the RIPE Atlas Network". In: *Free and Open Communications on the Internet*. USENIX, 2014. https://www.usenix.org/system/files/conference/foci14/foci14-anderson.pdf (cit. on p. 13).

[5] Ross J. Anderson. "The Eternity Service". In: *Theory and Applications of Cryptology*. CTU Publishing House, 1996, pp. 242–253. https://www.cl.cam.ac.uk/~rja14/Papers/eternity.pdf (cit. on p. 3).

[6] Yawning Angel and Philipp Winter. *obfs4 (The obfourscator)*. May 2014. https://gitweb.torproject.org/pluggable-transports/obfs4.git/tree/doc/obfs4-spec.txt (cit. on p. 15).

[7] Anonymous. "Towards a Comprehensive Picture of the Great Firewall's DNS Censorship". In: *Free and Open Communications on the Internet*. USENIX, 2014. https://www.usenix.org/system/files/conference/foci14/foci14-anonymous.pdf (cit. on pp. 10, 12).

[8] Simurgh Aryan, Homa Aryan, and J. Alex Halderman. "Internet Censorship in Iran: A First Look". In: *Free and Open Communications on the Internet*. USENIX, 2013. https://censorbib.nymity.ch/pdf/Aryan2013a.pdf (cit. on p. 13).

[9] *Big performance improvement for meek-azure*. tor-dev mailing list. Apr. 2015. https://lists.torproject.org/pipermail/tor-dev/2015-April/008637.html (cit. on p. 23).

[10] Arlo Breault, David Fifield, and George Kadianakis. *Registration over App Engine*. May 2013. https://bugs.torproject.org/8860 (cit. on p. 20).

[11] Willow Brugh. *San Francisco Hackathon/DiscoTech (+ RightsCon + Responsible Data Forum).* Mar. 2014. http://codesign.mit.edu/2014/03/sfdiscotech/ (cit. on p. 21).

[12] Cormac Callanan, Hein Dries-Ziekenheiner, Alberto Escudero-Pascual, and Robert Guerra. *Leaping Over the Firewall: A Review of Censorship Circumvention Tools.* Tech. rep. Freedom House, 2011. https://freedomhouse.org/report/special-reports/leaping-over-firewall-review-censorship-circumvention-tools (cit. on p. 29).

[13] Abdelberi Chaabane, Terence Chen, Mathieu Cunche, Emiliano De Cristofaro, Arik Friedman, and Mohamed Ali Kaafar. "Censorship in the Wild: Analyzing Internet Filtering in Syria". In: *Internet Measurement Conference.* ACM, 2014. http://conferences2.sigcomm.org/imc/2014/papers/p285.pdf (cit. on p. 14).

[14] Erinn Clark. *Tor Browser 3.6.4 and 4.0-alpha-1 are released.* The Tor Blog. Aug. 2014. https://blog.torproject.org/tor-browser-364-and-40-alpha-1-are-released (cit. on p. 22).

[15] Richard Clayton. "Failures in a Hybrid Content Blocking System". In: *Privacy Enhancing Technologies.* Springer, 2006, pp. 78–92. https://www.cl.cam.ac.uk/~rnc1/cleanfeed.pdf (cit. on pp. 7, 10).

[16] Richard Clayton, Steven J. Murdoch, and Robert N. M. Watson. "Ignoring the Great Firewall of China". In: *Privacy Enhancing Technologies.* Springer, 2006, pp. 20–35. https://www.cl.cam.ac.uk/~rnc1/ignoring.pdf (cit. on pp. 5, 7, 11).

[17] Jedidiah R. Crandall, Masashi Crete-Nishihata, and Jeffrey Knockel. "Forgive Us our SYNs: Technical and Ethical Considerations for Measuring Internet Filtering". In: *Ethics in Networked Systems Research.* ACM, 2015. http://ensr.oii.ox.ac.uk/wp-content/uploads/2015/07/Forgive-Us-Our-SYNs-Technical-and-Ethical-Considerations-for-Measuring-Internet-Censorship.pdf (cit. on p. 7).

[18] Jedidiah R. Crandall, Daniel Zinn, Michael Byrd, Earl Barr, and Rich East. "ConceptDoppler: A Weather Tracker for Internet Censorship". In: *Computer and Communications Security.* ACM, 2007, pp. 352–365. http://www.csd.uoc.gr/~hy558/papers/conceptdoppler.pdf (cit. on pp. 9, 12).

[19] Eric Cronin, Micah Sherr, and Matt Blaze. *The Eavesdropper's Dilemma.* Tech. rep. MS-CIS-05-24. Department of Computer and Information Science, University of Pennsylvania. http://www.crypto.com/papers/internet-tap.pdf (cit. on p. 5).

[20] Alberto Dainotti, Claudio Squarcella, Emile Aben, Kimberly C. Claffy, Marco Chiesa, Michele Russo, and Antonio Pescapé. "Analysis of Country-wide Internet Outages Caused by Censorship". In: *Internet Measurement Conference.* ACM, 2011, pp. 1–18. http://conferences.sigcomm.org/imc/2011/docs/p1.pdf (cit. on p. 13).

[21] Ronald Deibert, John G. Palfrey, Rafal Rohozinski, and Jonathan Zittrain, eds. *Access denied: the practice and policy of global Internet filtering.* Cambridge, Mass: MIT Press, 2008. ISBN: 978-0-262-54196-1. http://access.opennet.net/?page_id=61 (cit. on p. 11).

[22] Bill Dong. *Forbidden sites hijacked all over China.* Oct. 2002. https://web.archive.org/web/20021015162320/http://www.dit-inc.us/report/hj.htm (cit. on p. 7).

[23] Maximillian Dornseif. "Government mandated blocking of foreign Web content". In: *DFN-Arbeitstagung über Kommunikationsnetze*. Gesellschaft für Informatik, 2003, pp. 617–647. https://censorbib.nymity.ch/pdf/Dornseif2003a.pdf (cit. on p. 10).

[24] Eva Dou and Alistair Barr. *U.S. Cloud Providers Face Backlash From China's Censors*. Wall Street Journal. Mar. 2015. https://www.wsj.com/articles/u-s-cloud-providers-face-backlash-from-chinas-censors-1426541126 (cit. on p. 22).

[25] Frederick Douglas, Rorshach, Weiyang Pan, and Matthew Caesar. "Salmon: Robust Proxy Distribution for Censorship Circumvention". In: *Privacy Enhancing Technologies* 2016.4 (2016), pp. 4–20. https://www.degruyter.com/downloadpdf/j/popets.2016.2016.issue-4/popets-2016-0026/popets-2016-0026.xml (cit. on p. 6).

[26] Matthew Dunwoody. *APT29 Domain Fronting With TOR*. FireEye Threat Research Blog. Mar. 2017. https://www.fireeye.com/blog/threat-research/2017/03/apt29_domain_frontin.html (cit. on p. 25).

[27] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. "Protocol Misidentification Made Easy with Format-Transforming Encryption". In: *Computer and Communications Security*. ACM, 2013. https://eprint.iacr.org/2012/494.pdf (cit. on p. 15).

[28] Roya Ensafi, David Fifield, Philipp Winter, Nick Feamster, Nicholas Weaver, and Vern Paxson. "Examining How the Great Firewall Discovers Hidden Circumvention Servers". In: *Internet Measurement Conference*. ACM, 2015. http://conferences2.sigcomm.org/imc/2015/papers/p445.pdf (cit. on p. 16).

[29] Roya Ensafi, Philipp Winter, Abdullah Mueen, and Jedidiah R. Crandall. "Analyzing the Great Firewall of China Over Space and Time". In: *Privacy Enhancing Technologies* 2015.1 (2015). https://censorbib.nymity.ch/pdf/Ensafi2015a.pdf (cit. on p. 14).

[30] David Fifield. *A simple HTTP transport and big ideas*. tor-dev mailing list. Jan. 2014. https://lists.torproject.org/pipermail/tor-dev/2014-January/006159.html (cit. on p. 20).

[31] David Fifield. *GoAgent: Further notes on App Engine and speculation about a pluggable transport*. Tor Bug Tracker & Wiki. Oct. 2013. https://trac.torproject.org/projects/tor/wiki/doc/GoAgent?action=diff&version=2&old_version=1 (cit. on p. 20).

[32] David Fifield. *How to use the "meek" pluggable transport*. The Tor Blog. Aug. 2015. https://blog.torproject.org/how-use-meek-pluggable-transport (cit. on p. 22).

[33] David Fifield. *HOWTO use Lantern as a pluggable transport*. tor-dev mailing list. Mar. 2014. https://lists.torproject.org/pipermail/tor-dev/2014-March/006356.html (cit. on p. 21).

[34] David Fifield. *Outage of meek-azure*. tor-dev mailing list. Aug. 2015. https://lists.torproject.org/pipermail/tor-talk/2015-August/038780.html (cit. on p. 23).

[35] David Fifield. *Why the seeming correlation between flash proxy and meek on metrics graphs?* tor-dev mailing list. Sept. 2014. https://lists.torproject.org/pipermail/tor-dev/2014-September/007484.html (cit. on p. 22).

[36] David Fifield and Mia Gil Epner. *Fingerprintability of WebRTC*. Tech. rep. May 2016. https://arxiv.org/abs/1605.08805v1 (cit. on pp. 26, 28).

[37] David Fifield, Nate Hardison, Jonathan Ellithorpe, Emily Stark, Roger Dingledine, Phil Porras, and Dan Boneh. “Evading Censorship with Browser-Based Proxies”. In: *Privacy Enhancing Technologies Symposium*. Springer, 2012, pp. 239–258. http://crypto.stanford.edu/flashproxy/flashproxy.pdf (cit. on pp. 26, 27).

[38] David Fifield, Georg Koppen, and Klaus Layer. *Update the meek-amazon fingerprint to 09495296*. Oct. 2015. https://bugs.torproject.org/17473 (cit. on p. 23).

[39] David Fifield, Chang Lan, Rod Hynes, Percy Wegmann, and Vern Paxson. “Blocking-resistant communication through domain fronting”. In: *Privacy Enhancing Technologies* 2015.2 (2015). https://www.bamsoftware.com/papers/fronting/ (cit. on pp. 19, 21, 22, 26).

[40] David Fifield, Gabi Nakibly, and Dan Boneh. “OSS: Using Online Scanning Services for Censorship Circumvention”. In: *Privacy Enhancing Technologies Symposium*. Springer, 2013. http://freehaven.net/anonbib/papers/pets2013/paper_29.pdf (cit. on pp. 26, 28).

[41] David Fifield and Lynn Tsai. “Censors’ Delay in Blocking Circumvention Proxies”. In: *Free and Open Communications on the Internet*. USENIX, 2016. https://www.usenix.org/system/files/conference/foci16/foci16-paper-fifield.pdf (cit. on p. 16).

[42] Arturo Filastò and Jacob Appelbaum. “OONI: Open Observatory of Network Interference”. In: *Free and Open Communications on the Internet*. USENIX, 2012. https://www.usenix.org/system/files/conference/foci12/foci12-final12.pdf (cit. on pp. 7, 9, 17).

[43] Google. *Google Transparency Report: China, All Products, May 31, 2014–Present*. July 2014. https://www.google.com/transparencyreport/traffic/disruptions/124/ (cit. on p. 21).

[44] Google Cloud Platform. *Service Specific Terms*. Mar. 2015. https://web.archive.org/web/20150326000133/https://cloud.google.com/terms/service-terms (cit. on p. 23).

[45] Bennett Haselton. *List of possible weaknesses in systems to circumvent Internet censorship*. Nov. 2002. http://www.peacefire.org/circumventor/list-of-possible-weaknesses.html (cit. on p. 7).

[46] Amir Houmansadr, Giang T. K. Nguyen, Matthew Caesar, and Nikita Borisov. “Cirripede: Circumvention Infrastructure using Router Redirection with Plausible Deniability”. In: *Computer and Communications Security*. ACM, 2011, pp. 187–200. http://hatswitch.org/~nikita/papers/cirripede-ccs11.pdf (cit. on p. 15).

[47] Amir Houmansadr, Thomas Riedl, Nikita Borisov, and Andrew Singer. “I want my voice to be heard: IP over Voice-over-IP for unobservable censorship circumvention”. In: *Network and Distributed System Security*. The Internet Society, 2013. http://people.cs.umass.edu/~amir/papers/FreeWave.pdf (cit. on p. 15).

[48] *ICLab*. https://iclab.org/ (cit. on pp. 7, 17).

[49] Ben Jones, Roya Ensafi, Nick Feamster, Vern Paxson, and Nick Weaver. “Ethical Concerns for Censorship Measurement”. In: *Ethics in Networked Systems Research*. ACM, 2015. http://www.icir.org/vern/papers/censorship-meas.nsethics15.pdf (cit. on p. 7).

[50] George Kadianakis and Nick Mathewson. *obfs2 (The Twobfuscator)*. Jan. 2011. https://gitweb.torproject.org/pluggable-transports/obfsproxy.git/tree/doc/obfs2/obfs2-protocol-spec.txt (cit. on p. 15).

[51] George Kadianakis and Nick Mathewson. *obfs3 (The Threebfuscator)*. Jan. 2013. https://gitweb.torproject.org/pluggable-transports/obfsproxy.git/tree/doc/obfs3/obfs3-protocol-spec.txt (cit. on p. 15).

[52] Josh Karlin, Daniel Ellard, Alden W. Jackson, Christine E. Jones, Greg Lauer, David P. Mankins, and W. Timothy Strayer. “Decoy Routing: Toward Unblockable Internet Communication”. In: *Free and Open Communications on the Internet*. USENIX, 2011. http://static.usenix.org/event/foci11/tech/final_files/Karlin.pdf (cit. on p. 15).

[53] Sheharbano Khattak, Tariq Elahi, Laurent Simon, Colleen M. Swanson, Steven J. Murdoch, and Ian Goldberg. “SoK: Making Sense of Censorship Resistance Systems”. In: *Privacy Enhancing Technologies* 2016.4 (2016), pp. 37–61. https://www.degruyter.com/downloadpdf/j/popets.2016.2016.issue-4/popets-2016-0028/popets-2016-0028.xml (cit. on pp. 5, 8, 14).

[54] Sheharbano Khattak, Mobin Javed, Philip D. Anderson, and Vern Paxson. “Towards Illuminating a Censorship Monitor’s Model to Facilitate Evasion”. In: *Free and Open Communications on the Internet*. USENIX, 2013. https://censorbib.nymity.ch/pdf/Khattak2013a.pdf (cit. on pp. 7, 13).

[55] Karsten Loesing. *Counting daily bridge users*. Tech. rep. 2012-10-001. The Tor Project, Oct. 2012. https://research.torproject.org/techreports/counting-daily-bridge-users-2012-10-24.pdf (cit. on p. 19).

[56] Isis Agora Lovecruft and Henry de Valence. *HYPHAE: Social Secret Sharing*. Tech. rep. 2017-04-21. Apr. 2017. https://patternsinthevoid.net/hyphae/hyphae.pdf (cit. on p. 6).

[57] Graham Lowe, Patrick Winters, and Michael L. Marcus. *The Great DNS Wall of China*. Tech. rep. New York University, 2007. https://cs.nyu.edu/~pcw216/work/nds/final.pdf (cit. on p. 12).

[58] Mohammad Mahdian. “Fighting Censorship with Algorithms”. In: *International Conference on Fun with Algorithms*. Springer, 2010, pp. 296–306. http://mahdian.org/censorship.pdf (cit. on p. 6).

[59] Bill Marczak, Nicholas Weaver, Jakub Dalek, Roya Ensafi, David Fifield, Sarah McKune, Arn Rey, John Scott-Railton, Ron Deibert, and Vern Paxson. “An Analysis of China’s ‘Great Cannon’”. In: *Free and Open Communications on the Internet*. USENIX, 2015. https://www.usenix.org/system/files/conference/foci15/foci15-paper-marczak.pdf (cit. on p. 14).

[60] Bill Marczak, Nicholas Weaver, Jakub Dalek, Roya Ensafi, David Fifield, Sarah McKune, Arn Rey, John Scott-Railton, Ron Deibert, and Vern Paxson. *China's Great Cannon*. Tech. rep. The Citizen Lab, Apr. 2015. https://citizenlab.ca/2015/04/chinas-great-cannon/ (cit. on p. 22).

[61] Damon McCoy, Jose Andre Morales, and Kirill Levchenko. "Proximax: A Measurement Based System for Proxies Dissemination". In: *Financial Cryptography and Data Security*. Springer, 2011. http://cseweb.ucsd.edu/~klevchen/mml-fc11.pdf (cit. on p. 6).

[62] *meek*. Tor Bug Tracker & Wiki. https://trac.torproject.org/projects/tor/wiki/doc/meek (cit. on pp. 22, 23).

[63] Zubair Nabi. "The Anatomy of Web Censorship in Pakistan". In: *Free and Open Communications on the Internet*. USENIX, 2013. https://censorbib.nymity.ch/pdf/Nabi2013a.pdf (cit. on p. 14).

[64] Milad Nasr, Sadegh Farhang, Amir Houmansadr, and Jens Grossklags. *Enemy At the Gateways: A Game Theoretic Approach to Proxy Distribution*. Tech. rep. Sept. 2017. https://arxiv.org/abs/1709.04030v1 (cit. on p. 6).

[65] OpenNet Initiative. *Internet Filtering in China in 2004-2005: A Country Study*. https://opennet.net/studies/china (cit. on p. 11).

[66] Jong Chun Park and Jedidiah R. Crandall. "Empirical Study of a National-Scale Distributed Intrusion Detection System: Backbone-Level Filtering of HTML Responses in China". In: *Distributed Computing Systems*. IEEE, 2010, pp. 315–326. https://www.cs.unm.edu/~crandall/icdcs2010.pdf (cit. on p. 12).

[67] Mike Perry. *Tor Browser 4.0 is released*. The Tor Blog. Oct. 2014. https://blog.torproject.org/tor-browser-40-released (cit. on p. 22).

[68] Andreas Pfitzmann and Marit Hansen. *A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management*. Version 0.34. Aug. 2010. https://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.34.pdf (cit. on p. 5).

[69] Matthew Prince. Hacker News comment. Mar. 2015. https://news.ycombinator.com/item?id=9234367 (cit. on p. 22).

[70] Hal Roberts, Ethan Zuckerman, and John Palfrey. *2011 Circumvention Tool Evaluation*. Tech. rep. Berkman Center for Internet and Society, Aug. 2011. https://cyber.law.harvard.edu/publications/2011/2011_Circumvention_Tool_Evaluation (cit. on p. 29).

[71] David Robinson, Harlan Yu, and Anne An. *Collateral Freedom: A Snapshot of Chinese Internet Users Circumventing Censorship*. Apr. 2013. https://www.opentech.fund/article/collateral-freedom-snapshot-chinese-users-circumventing-censorship (cit. on p. 20).

[72] Andreas Sfakianakis, Elias Athanasopoulos, and Sotiris Ioannidis. "CensMon: A Web Censorship Monitor". In: *Free and Open Communications on the Internet*. USENIX, 2011. http://static.usenix.org/event/foci11/tech/final_files/Sfakianakis.pdf (cit. on pp. 9, 12).

[73] Charlie Smith. GreatFire. Mar. 2015. https://en.greatfire.org/blog/2015/mar/we-are-under-attack (cit. on p. 22).

[74] *Snowflake*. Tor Bug Tracker & Wiki. https://trac.torproject.org/projects/tor/wiki/doc/Snowflake (cit. on pp. 26, 28).

[75] Yair Sovran, Jinyang Li, and Lakshminarayanan Subramanian. *Unblocking the Internet: Social Networks Stymie Censors*. Tech. rep. TR2008-918. New York University, 2008. http://kscope.news.cs.nyu.edu/pub/TR-2008-918.pdf (cit. on p. 6).

[76] Yair Sovran, Alana Libonati, and Jinyang Li. "Pass it on: Social Networks Stymie Censors". In: *International Workshop on Peer-to-Peer Systems*. USENIX, 2008. http://www.cs.toronto.edu/iptps2008/final/73.pdf (cit. on p. 6).

[77] *Toosheh*. http://www.toosheh.org/en.html (cit. on p. 5).

[78] Michael Carl Tschantz, Sadia Afroz, Anonymous, and Vern Paxson. "SoK: Towards Grounding Censorship Circumvention in Empiricism". In: *Symposium on Security & Privacy*. IEEE, 2016. https://www.eecs.berkeley.edu/~sa499/papers/oakland2016.pdf (cit. on pp. 5, 9, 14, 29).

[79] *uProxy*. https://www.uproxy.org/ (cit. on p. 28).

[80] John-Paul Verkamp and Minaxi Gupta. "Inferring Mechanics of Web Censorship Around the World". In: *Free and Open Communications on the Internet*. USENIX, 2012. https://www.usenix.org/system/files/conference/foci12/foci12-final1.pdf (cit. on p. 13).

[81] Qiyan Wang, Zi Lin, Nikita Borisov, and Nicholas J. Hopper. "rBridge: User Reputation based Tor Bridge Distribution with Privacy Preservation". In: *Network and Distributed System Security*. The Internet Society, 2013. http://www-users.cs.umn.edu/~hopper/rbridge_ndss13.pdf (cit. on p. 6).

[82] Zachary Weinberg, Jeffrey Wang, Vinod Yegneswaran, Linda Briesemeister, Steven Cheung, Frank Wang, and Dan Boneh. "StegoTorus: A Camouflage Proxy for the Tor Anonymity System". In: *Computer and Communications Security*. ACM, 2012. http://www.frankwang.org/papers/ccs2012.pdf (cit. on p. 15).

[83] Philipp Winter. "Measuring and circumventing Internet censorship". PhD thesis. Karlstad University, 2014 (cit. on p. 5).

[84] Philipp Winter and Stefan Lindskog. "How the Great Firewall of China is Blocking Tor". In: *Free and Open Communications on the Internet*. USENIX, 2012. https://www.usenix.org/system/files/conference/foci12/foci12-final2.pdf (cit. on p. 12).

[85] Sebastian Wolfgarten. "Investigating large-scale Internet content filtering". MA thesis. Dublin City University, Aug. 2006. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.133.5778&rep=rep1&type=pdf (cit. on p. 12).

[86] Joss Wright. *Regional Variation in Chinese Internet Filtering*. Tech. rep. University of Oxford, 2012. https://papers.ssrn.com/sol3/Delivery.cfm/SSRN_ID2265775_code1448244.pdf?abstractid=2265775&mirid=3 (cit. on p. 12).

[87] Joss Wright, Tulio de Souza, and Ian Brown. “Fine-Grained Censorship Mapping: Information Sources, Legality and Ethics”. In: *Free and Open Communications on the Internet*. USENIX, 2011. http://static.usenix.org/event/foci11/tech/final_files/Wright.pdf (cit. on pp. 2, 7).

[88] Eric Wustrow, Scott Wolchok, Ian Goldberg, and J. Alex Halderman. “Telex: Anticensorship in the Network Infrastructure”. In: *USENIX Security Symposium*. USENIX, 2011. https://www.usenix.org/event/sec11/tech/full_papers/Wustrow.pdf (cit. on p. 15).

[89] Xueyang Xu, Z. Morley Mao, and J. Alex Halderman. “Internet Censorship in China: Where Does the Filtering Occur?” In: *Passive and Active Measurement Conference*. Springer, 2011, pp. 133–142. https://web.eecs.umich.edu/~zmao/Papers/china-censorship-pam11.pdf (cit. on p. 12).