EMPOWERING BYSTANDERS TO FACILITATE INTERNET CENSORSHIP MEASUREMENT AND CIRCUMVENTION

A Thesis
Presented to
The Academic Faculty

by

Samuel Read Burnett

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computer Science

Georgia Institute of Technology
August 2014

EMPOWERING BYSTANDERS TO FACILITATE INTERNET CENSORSHIP MEASUREMENT AND CIRCUMVENTION

ACKNOWLEDGEMENTS

Many people have contributed to this thesis, either through direct collaboration, feedback, and inspiration, or by providing a supportive environment which made my life as a graduate student much more enjoyable and productive.

My doctoral advisor, Nick Feamster, has had a tremendous impact on my life over the past six years. He very much leads by example, and his boundless energy, enthusiasm, and optimism has inspired both me and my fellow students. Nick has exemplified all of the characteristics of a good mentor: he helped me select a research topic that both fits my interests and is very relevant to the academic community; he developed contacts with key figures in research and industry, both ensuring my work always has an audience and boosting my career prospects; he guided me through the often frustrating process of writing research papers, submitting them for publication, and presenting them for an audience; and he ensured I never had to worry about funding. In addition, Nick has instilled in me the importance of good communication, and has done an excellent job of showing me how normally introverted and mild-mannered people like myself can still be effective communicators and leaders.

The other members of my thesis committee, Mustaque Ahamad, Mostafa Ammar, Phillipa Gill, and Wenke Lee, gave excellent feedback and a fresh perspective on the work in this dissertation. Special thanks to Phillipa for giving detailed suggestions and comments on earlier versions of this dissertation, and to Wenke for giving feedback at many points throughout my time at Georgia Tech.

I have been fortunate to work with a wide array of supportive, intelligent, and wise callaborators and coauthors. Mukarram Tariq, Matt Braithwaite, Luke Sandberg, and Michael Davidson, my internship mentors and teammates at Google, helped shape my choice of career and paved my way toward achieving that goal. My internship with Ben Greenstein and David Wetherall at Intel Labs Seattle was both very instructive and fun. Thank you to Phillipa Gill, Sathya Gunasekaran, Ben Jones, and the many others from Stonybrook and the Citizen Lab who attended weekly censorship measurement meetings during my final year for inspiring me to write Chapter 3 of this thesis. Thank you to Meredith Whittaker for organizing two fantastic Internet censorship measurement workshops, and to all the attendees of those workshops for providing valuable insight and feedback on my work.

Although it does not feature heavily in this dissertation, I invested much time and effort in helping to build BISmark, a testbed of home routers for conducting home network measurements. Working on BISmark has shaped how I approach systems research and will surely benefit me greatly in my career ahead. Thank you to Srikanth Sundaresan, Steve Woodrow, Walter de Donato, and the many BISmark contributors, developers, and users.

Life in graduate school would have been quite boring had I not had the pleasure of sharing it with some amazing colleagues. Many thanks to Ahmed, Arpit, Ben, Bilal, Chaitrali, Cristian, Giuseppe, Joon, Maria, Miseon, Murtaza, Robert, Saamer, Samantha, Sathya, Shuang, Srikanth, Steve, Valas, Walter, Yogesh, and everyone else from the NTG lab for great lunchtime conversations, insightful and honest feedback on papers and presentations, commiserating on life as grad students, running a half marathon, attending concerts at the Atlanta Symphony, tolerating my lack of culinary imagination by consuming countless pizzas and apple pies, showing me that Peking duck is much tastier than turkey even on Thanksgiving, and joining me for "coffee" breaks that very rarely resulted in me actually ordering coffee.

Ron Mendola, Pete Ciaschini, Brandon Houghtalen, Andrea Brown, and the hundreds of members of the Georgia Tech Symphony Orchestra enabled me to escape research for a few hours each week and continue to pursue my passion for classical music. I owe them for my relative sanity during grad school. I am eternally indebted to Margaret Hopkins, my violin teacher since childhood, for introducing me to the world of music and inspiring me to continue making it an integral part of my life.

My best and most loyal friend, Jack, has been great company and a constant source of perspective, insight, advice, and humor over the past ten years. Finally, I share my past, present, and future successes with my parents, Lee and Gail, and my sister, Emma, who continue to be to most unconditionally supportive, nurturing, and loving family I could ever hope for.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Free and open exchange of information on the Internet is at risk: more than 60 countries practice some form of Internet censorship, and both the number of countries practicing censorship and the proportion of Internet users who are subject to it are on the rise. Understanding and mitigating these threats to Internet freedom is a continuous technological arms race between security researchers and advocates, and many of the most influential governments and corporations.

By its very nature, Internet censorship varies drastically from region to region, which has impeded nearly all efforts to observe and fight it on a global scale. Researchers and developers in one country may find it very difficult to study censorship in another; this is particularly true for those in North America and Europe attempting to study notoriously pervasive censorship in Asia and the Middle East.

This dissertation develops techniques and systems that empower users in one country, or *bystanders*, to assist in the measurement and circumvention of Internet censorship in another. Our work builds from the observation that there are people *everywhere* who are willing to help us if only they knew how. First, we develop Encore, which allows webmasters to help study Web censorship by collecting measurements from their sites' visitors. Encore leverages weaknesses in cross-origin security policy to collect measurements from a far more diverse set of vantage points than previously possible. Second, we build Collage, a technique that uses the pervasiveness and scalability of user-generated content to disseminate censored content. Collage's novel communication model is robust against censorship that is significantly more powerful than governments use today. Encore and Collage make

it significantly easier for people everywhere to help study and circumvent Internet censorship.

CHAPTER I

INTRODUCTION

The past twenty-five years have seen the Internet grow from humble beginnings as a research and education network operated by the United States government to a multi-purpose, multi-national network spanning all countries and continents. Although the Internet's development has transformed commerce, education, entertainment, and nearly every other facet of modern life, an increasing number of governments have sought to restrict how their citizens use it. Several studies have reported that dozens of countries restrict Internet communications in some way, and it is likely that far more entities manipulate content or communications in some fashion [69, 111]. Both the number of countries that restrict Internet access and the number of Internet users affected by those restrictions are likely to continue to increase as more countries with historically repressive governments become better connected. We refer to these restrictions on Internet usage and services as *Internet censorship*.

Governments and organizations enforce Internet censorship in a variety of ways. Sometimes they use legal or extralegal tactics to restrict information at its source. For example, China is notorious for requiring search engines and blogging platforms operating within its borders to remove offensive content [8, 75], while several governments intimidate or persecute journalists and bloggers when they see fit [33, 95, 129]. More subtle mechanisms may compel users and organizations to filter their own communications without explicit orders to do so, *e.g.*, by surveilling instant message conversations, as TOM-Skype and LINE do in China [40, 81]; or threatening expulsion of foreign media agencies [54].

Directly censoring producers of information in these ways is expensive, unscalable, and imperfect. Not only must governments monitor potentially censored services to ensure proper enforcement, they might find it difficult to censor individuals and organizations outside their jurisdiction. Instead, many governments censor information en route to their citizens by disrupting the network traffic to or from specific domains, sites, or services. China, Iran, Pakistan, Turkey, Syria, and dozens of other governments have implemented systems that selectively block Internet traffic from reaching its intended destination [77, 111]. A few examples: Turkey briefly prevented access to Twitter and YouTube prior to elections in 2014 [146]; Pakistan has blocked YouTube since 2012 for hosting religiously offensive videos [117]; and China recently censored the Web sites of several foreign news agencies

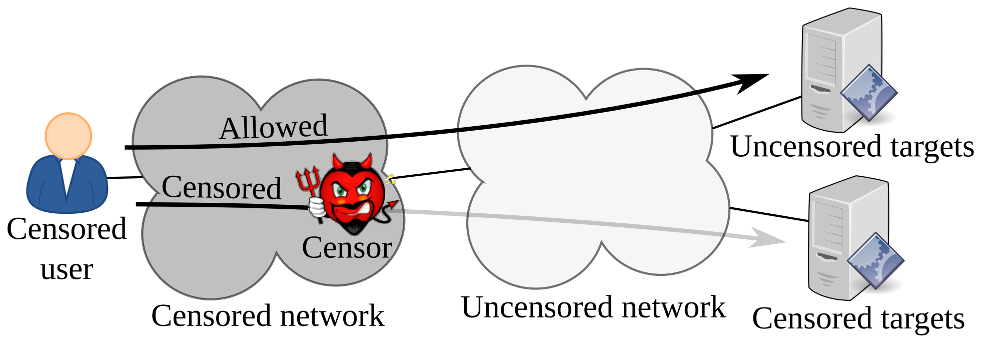


Figure 1: A censor restricts communication on the censored network and disrupts attempts to access some users, sites, or services (*i.e.*, *targets*). Censorship may vary over time and by location.

after they exposed corruption among government officials [21]. In cases where governments perceive outright blocking to be too drastic, they may instead degrade performance to make communication inconvenient rather than impossible; China and Iran have used this technique to great effect [78,89]. On the other end of the spectrum, some governments have resorted to complete disconnection from the Internet in cases of extreme political unrest, as Egypt and Libya did during the Arab Spring in 2011 [38,41].

This dissertation focuses on this latter class of Internet censorship — disruption of network traffic destined for specific domains, sites, or services — which we refer to as *network censorship*. Figure 1 shows the general setting. Users inside a country that employs network censorship try to communicate with other users, sites, and services, and the censor attempts to disrupt access to some subset of these destinations. Network censorship mechanisms are among the most prevalent way to enforce censorship worldwide because of both their effectiveness and their ease of implementation, and they today impact Internet users in many countries. Fortunately, these mechanisms are also the easiest to study because disruption of network traffic tends to be relatively obvious, lending itself to technical analysis and development of countermeasures. The remainder of this dissertation only considers network censorship except when otherwise noted, and often uses the terms "network censorship" and "Internet censorship" interchangeably.

Governments most commonly use network censorship to restrict access to the Web (*i.e.*, HTTP and HTTPS). Figure 2 illustrates many ways that they may censor an HTTP(S) connection or its requisite DNS lookup, including hijacking DNS requests, reseting TCP connections to certain IP addresses, redirecting HTTP requests, blocking entire protocols or port numbers (*e.g.*, port 443 for HTTPS), and restricting entire IP prefixes and autonomous systems by modifying BGP configuration. Each mechanism operates at a certain granularity and with a certain efficiency, dictated by where in the network stack that technique

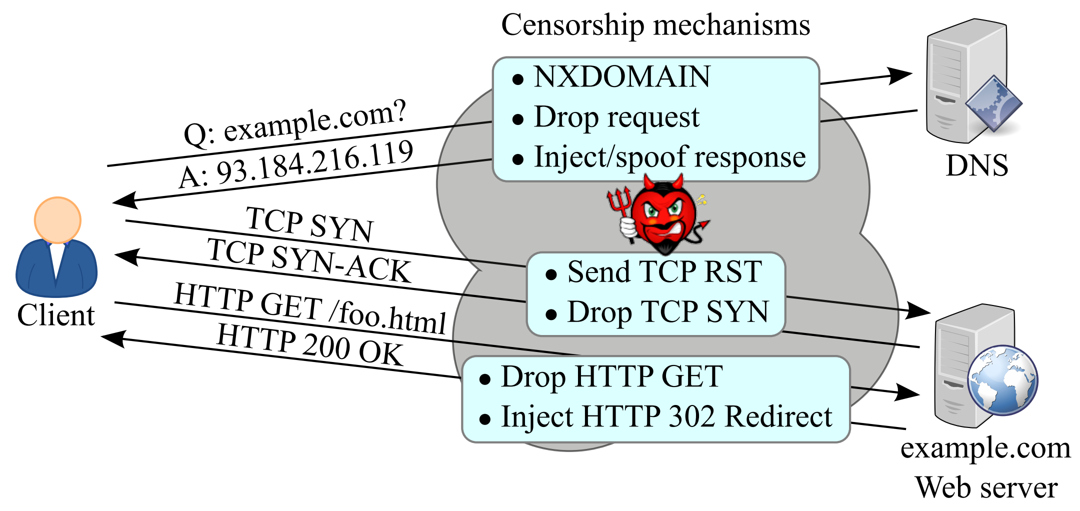


Figure 2: Censors may restrict access to `http://example.com/foo.html` by disrupting each each stage of an HTTP connection using several mechanisms.

operates. For example, governments and ISPs can often implement censorship very efficiently by hijacking DNS queries in ISP recursive resolvers, but this mechanism can only operate at the level of domain names and cannot censor individual services or applications without inflicting collateral damage on others. We use examples of Web censorship throughout this dissertation.

1.1 Studying network censorship

This dissertation develops new techniques for censorship discovery, measurement, and circumvention. We introduce these concepts now and elaborate on them in Chapter 2.

Discovering censorship. Service operators, researchers, activists, and users all seek to quickly and accurately discover when a government starts censoring a site or service, both to understand the scope and dynamics of censorship and to quickly deploy countermeasures. Without good censorship discovery, service operators often only learn about censorship of their service through word of mouth or press reports, users have difficulty distinguishing between censorship and other kinds of network failure, and researchers and activists lack fast and consistent ways to identify new cases of censorship and deploy countermeasures.

Measuring censorship. After discovering that a particular site or service is censored, we often wish to measure more details about where, when, and how the censorship is implemented. Quantities to measure include the duration of censorship, the technical mechanism used to enforce it, and whether censorship is consistent across regions and ISPs. These measurements can inform researchers and activists how best to target and develop new circumvention techniques, and can help public policy experts mold future Internet policy.

Circumventing censorship. Lastly, once we understand what, where, when, and how sites and services are censored, researchers, activists, and users all seek ways to circumvent this censorship. Likewise, governments and security experts pursue understanding of the limitations and possible weaknesses of censorship mechanisms. Censorship circumvention is an arms race; governments impose ever more sophisticated and nuanced censorship mechanisms, while researchers and activists develop countermeasures to these mechanisms.

1.2 Fundamental challenges of studying network censorship

Although researchers face numerous obstacles to studying censorship discovery, measurement, and circumvention, two major challenges have dominated all work in this space. They motivate the novel techniques and systems that we develop in this dissertation.

Network censorship mechanisms are diverse, dynamic, and unknown. The example mechanisms in Figure 2 only scratch the surface of the diversity of mechanisms that governments employ to enforce censorship, which stymies attempts to comprehensively measure or circumvent it. Researchers can rarely be sure that their measurements are complete, or that the mechanisms they observe behave in standard or predictable ways. Developers of circumvention tools must compete against censors that continually deploy ever more sophisticated and diverse censorship mechanisms.

This diversity stems from Internet censorship's origins. Censorship is not inherently technical, even when applied to sophisticated technical systems like the Internet. Governments and organizations have sought to control access to information throughout human history, and much of the responsibility for overcoming such censorship lies in the hands of non-technical experts in public policy, law, and activism to guide governments away from repressive policies. Internet censorship is thus born of politics, not technology, and these political origins manifest themselves in technology in many ways. Put another way, governments often strive for certain policy goals without regard for standard technical procedures to accomplish them, leading to diverse implementations. Governments and service providers may enforce similar censorship policies (*i.e.*, to restrict the flow of information on the Internet), yet share little common technical basis for doing so.

Despite these non-technical origins, researchers and activists have nonetheless developed a fragmented arsenal of technology to study and circumvent the various manifestations of Internet censorship that governments devise. Circumvention techniques range from simple proxy servers and virtual private networks (VPNs) to more sophisticated anonymity networks and covert communication channels. They are often tailored to the needs of users in individual countries. Censors and citizens engage in an ongoing arms race to develop more powerful censorship mechanisms and circumvention tools, and this arms race is nowhere near finished.

Censorship is regional. Although the arms race for censorship mechanisms and circumvention tools will continue into the foreseeable future, researchers face an uphill battle, as it is not enough to simply design and implement state-of-art censorship circumvention and measurements tools; these tools must make it into the hands of the users who need them in censored countries across the globe. Even though researchers have already developed circumvention techniques that evade every widely-deployed censorship scheme, only a very small fraction of users actually know how to install and use these tools. The situation is even worse for researchers wishing to measure and document censorship mechanisms and worldwide censorship trends. Measuring properties of Internet censorship mechanisms is not a technically difficult task in most cases, but there has been no global, systematic measurement study of it to date; this is largely because, although it is very easy to write censorship measurement tools, it is extremely difficult to deploy them in areas of the world that experience the most censorship.

Although many censorship mechanisms are simply applications of existing network security techniques, these new applications often yield interesting research problems because of their new context. These new contexts mean that, unlike most research in computer networking (and computer science in general), work on Internet censorship must cross many language, culture, and geographic barriers before it is most useful. Researchers and developers are most often located in North American and European countries that, while certainly not without their own problems, don't experience the same level of censorship as elsewhere in the world.

1.3 Bridging the gap between censored and uncensored users

Thesis statement. This dissertation counteracts Internet censorship's diversity and regionalism by narrowing the gap between users in censored and uncensored regions. We demonstrate that parties in uncensored regions, or *bystanders*, can help measure and circumvent the censorship experienced by users in other regions. Our work builds on the observation that there are bystanders *everywhere* who are eager to help discover, measure, and circumvent Internet censorship if only they knew how. We develop novel techniques and systems that empower these bystanders to do so.

1.4 Contributions

This dissertation makes the following contributions in defense of the thesis statement:

1. *Encore, a system that lets Webmasters help measure Web censorship.* We design and implement Encore, the first system that measures Internet censorship from unmodified Web browsers. Doing so enables Encore to collect measurements of Web censorship from far more vantage points than previously possible, thereby chipping away at censorship's endemic regionalism. Encore demonstrates that bystanders who operate Web sites in uncensored regions can help measure Internet censorship.

2. *A technique for circumventing censorship with user-generated content.* We explore strong methods for censorship circumvention with Collage, an automated technique for disseminating censored data by hiding it amongst legitimate content across the Web. Collage's security properties counteract diversity of censorship because they enable communication in the presence of a broad range of current and future adversaries, and will be particularly valuable in the fight future censors that could be significantly more powerful than those today. Collage success depends on bystanders who either operate user-generated content hosting infrastructure or are users of these services to help circumvent Internet censorship.

We now elaborate on these contributions.

An opportunity for Webmasters to help measure censorship. We design and implement Encore, the first censorship measurement effort that enables bystander Webmasters to assist in gathering measurements. After adding a single line of HTML code to a Web site, every visitor to that site will contribute data about how it experiences censorship. Encore counteracts Internet censorship's regionalism; rather than requiring researchers to cross language and culture barriers to deploy custom measurement code on clients in each censored country, Encore lets researchers instead recruit a relatively small number of Webmasters anywhere in the world.

Measuring censorship from unmodified Web browsers. Encore is also the first system that collects measurements of censorship from unmodified Web browsers. Although well-established Web security mechanisms prevent sites from accessing resources hosted by arbitrary third parties, Encore skirts these mechanisms and allows researchers to collect censorship measurements without custom browser extensions or plugins. This vastly improves Encore's potential deployment footprint over browser-based measurement tools that require such modifications [49]. After less than one month of deployment on a small handful of Web sites, Encore has already collected measurements from **X** users in **Y** countries.

Robust communication via user-generated content hosts. We develop Collage, a technique for distributing covert messages via user-generated content hosts. Collage hides these messages inside ordinary content (*e.g.*, photos, videos, etc) hosted on user-generated sites like Flickr, YouTube, Picasa, and Twitter. To cope with reliability and capacity problems, Collage applies information theoretic techniques and algorithms in a novel setting, resulting in a robust communication channel that is resistant to a censor's attempts to block or manipulate significant fractions of its cover media. Collage counteracts Internet censorship's diversity with a powerful new circumvention model that is effective against a broad range of censorship mechanisms, and demonstrates that both user-generated content hosts and their users are bystanders that can help circumvent Internet censorship.

Censorship circumvention without dedicated infrastructure. Circumvention tools typically require some kind of dedicated infrastructure. For example, Tor requires a network of relay nodes. Collage leverages existing infrastructure, which makes it more difficult for governments to disrupt Collage without incurring significant collateral damage on unrelated services.

The first *hide-within* circumvention tool. Designers of circumvention tools typically try to achieve covertness by mimicking the network traffic generated by other, popular applications; unfortunately, recent research has demonstrated that these so-called *parrot* circumvention systems almost always contain significant security flaws because it's very difficult to perfectly mimic every aspect of a real application's traffic [84]. In contrast, Collage is the first *hide-within* censorship circumvention tool; this new class of tools tunnels its communications inside the traffic of a real, running instance of a popular application (*e.g.*, Firefox) rather than trying to mimic that application's traffic. Hide-within circumvention tools are significantly more robust against active and passive attacks.

1.5 Outline

We present background on network censorship, censorship measurement, and censorship circumvention in Chapter 2. Chapter 3 presents Encore, a new technique and system for measuring Internet censorship from unmodified Web browsers. Chapter 4 presents Collage, a circumvention technique that enables bystanders to help measure Internet censorship. We conclude with a discussion of general lessons we learned while designing these two systems in Chapter 5.

1.6 Bibliographic notes

Collage appeared as a poster at NSDI 2009 [24], a full paper at USENIX Security 2010 [25], and a demo at SIGCOMM 2010 [26]. Our position paper summarizing the difficulties of measuring Internet censorship inspired us to build Encore [23].

CHAPTER II

BACKGROUND AND RELATED WORK

This chapter introduces network censorship in more detail, focusing on challenges and existing techniques for measuring and circumventing it. We begin with an overview of the mechanisms that governments use to enforce censorship, then discuss approaches that researchers have taken to study it.

We cover background material and related work in a single chapter because the two are very closely related when studying Internet censorship. For most researchers, particularly in the western world, censorship is an emergent behavior of the Internet ecosystem; with only a few exceptions, our knowledge of it is drawn almost exclusively from observations and measurements rather than formal systems design, implementation, and evaluation. Thus, discussing background material without simultaneously discussing the studies and measurements that inform that material makes little sense.

2.1 Technical censorship mechanisms

This dissertation only considers *network censorship*, in which governments, ISPs, or organizations disrupt network traffic to subsets of users, domains, IP addresses, URLs, or services. Figure 1 shows the general problem setting. For convenience, we often refer to these governments, ISPs, or organizations as *censors*, and members of subsets of users, domains, IP address, URLs, or services collectively as *censorship targets* or simply *targets*.

This section introduces the technical *mechanisms* that censors use to enforce censorship and that we aim to measure and circumvent in Chapters 3 and 4. Given the inherently regional and secretive nature of censorship, our knowledge about how governments implement censorship policies may be inaccurate or incomplete, and only reflects information gleaned from experimental research. We defer discussion of how researchers uncovered and study these mechanisms to Section 2.4.

Censors can impose network censorship through nearly every piece of network infrastructure and at all layers of the network stack: they disrupt BGP connectivity, DNS resolution, TCP connection establishment, HTTP requests and responses, entire application protocols, and more. Most mechanisms aren't specific to censorship enforcement; rather,

Table 1: Censors can employ many mechanisms at each layer of network infrastructure and the network stack. Generally speaking, as scope becomes more specific, complexity and cost of enforcement increases. This table is not exhaustive (nor could it be); it merely gives examples of censorship mechanisms observed somewhere in the world.

Scope of censorship	Example mechanisms
Entire AS	Withdraw BGP advertisements Physically disconnect cables
Domain name	Install new rules on ISP recursive resolvers Drop DNS requests to certain domains Inject DNS responses certain domains
IP address	Drop packets destined for an IP address Inject TCP RST upon SYN to an IP address
HTTP URL	Drop HTTP request Return alternate HTTP response

↑ Greater scope

Greater cost and complexity ↓

they are applications of traditional network security techniques in unconventional environments. We can parameterize these mechanisms along several axes:

Scope and specificity. Some mechanisms are capable of disrupting access to individual resources, while others inherently impact a much larger set of (possibly unrelated) resources. For example, mechanisms that disrupt DNS cannot disrupt access to individual URLs, and mechanisms that tamper with BGP can only restrict access to entire IP prefixes or autonomous systems (ASes). This was the case in Pakistan, which blocked the entirety of YouTube in September, 2012 over only a handful of religiously offensive videos, simply because the country's censorship infrastructure at the time was incapable of restricting access to individual URLs [105]; Syria, by contrast, often uses extremely specific filtering rules to pinpoint censorship to individual resources without causing much collateral damage [3]. Table 1 summarizes a few common scopes of censorship and some mechanisms a censor could use to enforce them. More generally, the scope of censorship can often indicate the mechanism (or class of mechanisms) of censorship; we exploit this fact in Section 3.3 to infer mechanisms of Web censorship.

Note that countries sometimes want to disrupt access to entire services or domains even when they have the capacity for more targeted disruption. For example, China often blocks entire domains even though its firewall is sophisticated enough to target individual URLs and keywords [77].

Censor by address vs by content. Censors typically use one of two broad criteria to decide whether to restrict communication. The first is by addresses that identify communication endpoints; many governments censor specific domains or URLs, which are both types of addresses. The second is by the actual content of the communication; for example, censors like China filter communications that contain certain keywords. A useful way to differentiate between these kinds of censorship is that address-based censorship restricts who you talk to, while content-based censorship restricts what you talk about. This distinction is important because circumvention techniques are often quite different; circumvention of address-based censorship focuses on concealing traffic's true destination by routing it through an intermediary, whereas content-based censorship often tries to obfuscate the contents itself. Section 2.5 discusses these ideas further.

Complexity and cost. Mechanisms that target higher layers of the network stack are often more complex and, therefore, more expensive. This is often because higher layers require the mechanism to maintain state across multiple packets or flows. For example, blocking access to an entire domain name requires no state, but censoring individual URLs may require a censor to reassemble TCP streams. We often see this cost reflected in the shared chronology of censorship across different countries: countries often implement DNS-based censorship first because of its simplicity and efficiency, then migrate to more precise techniques later to increase flexibility. An important property of Collage (Chapter 4) is that, while it certainly cannot defeat all forms of censorship, the forms of censorship that it doesn't defeat would be very expensive and complicated to implement.

Transparency. Many countries display an informational Web page to users who try to access a censored resource; others disrupt connections without explanation [153]. Mechanisms that don't give explicit feedback about the existence of censorship may make it difficult for users to distinguish between censorship and other kinds of network problems. On the other hand, automatically distinguishing between block pages and legitimate content can be surprisingly difficult in general because pages can vary dramatically over time and by region.

On-path vs off-path. Newer, stateful mechanisms often operate on-path, disrupting flows as they traverse censorship middleboxes using deep packet inspection (DPI). Other techniques operate off-path. For example, because China's DNS and TCP censorship injects additional packets into a stream rather than dropping or modifying existing packets, they can operate off-path for greater efficiency. Off-path techniques are more restricted and may be easier to circumvent. For example, off-path techniques can only inject additional packets and cannot modify or remove existing traffic; Clayton *et al.* [35] showed how to circumvent Chinese censorship by ignoring packets injected by its off-path censorship mechanism.

Active vs passive. Most censorship is passive, in the sense that censors do not attempt to probe client machines or services before deciding to censor them. China, however, sends active probes to suspected Tor bridge nodes before blocking them to confirm whether they speak the Tor protocol [159]. At the time of writing, researchers have focused most attention on measuring and circumventing passive techniques, and these efforts rarely carry over to active mechanisms; thus, most circumvention tools don't yet defend against active mechanisms. On the other hand, active mechanisms are easier to detect and observe in some instances (*e.g.*, researchers can identify probes from the Chinese government to suspected Tor bridge nodes). One notable feature of Collage (Chapter 4) is that it, unlike many other circumvention tools, is inherently resistant to many active probing attacks.

Commercial vs custom. Some countries (*e.g.*, China) build their own censorship infrastructure from scratch, while many others purchase off-the-shelf Deep Packet Inspection (DPI) equipment from multinational corporations. Commercial solutions are sometimes easier to study because we can share measurement and circumvention techniques across all countries that use the same equipment [42].

Centralized vs distributed. Smaller countries tend to centralize their censorship infrastructure by placing it on a national backbone, while larger countries delegate responsibility for censorship to each ISP. Still others do both. Censorship measurement schemes often aim to quantify the differences in censorship enforcement between ISPs in the same country [105, 165].

2.2 The network censorship arms race

Researchers study three closely-related components of the network censorship ecosystem: *discovery* decides what targets to measure for censorship; *measurement* conducts experiments and collects data about what, where, when, and how targets are censored and the mechanisms that enforce this censorship; and *circumvention* counteracts censorship by finding and exploiting weaknesses in censorship mechanisms.

Figure 3 illustrates that censors and researchers engage in an arms race to build more powerful censorship and circumvention tools. Researchers discover and measure instances of censorship, then use this knowledge to build circumvention tools. Meanwhile, censors observe these circumvention tools and deploy new censorship mechanisms to thwart them, which may cause blocking of additional resources. Researchers then observe these changes in the censor's strategy by collecting more measurements, and the arms race continues *ad infinitum*. This arms race ensures that the network censorship research landscape is always changing and means that although we do our best to design generic, robust measurement and circumvention tools, it is difficult to predict whether the discovery, measurement, and

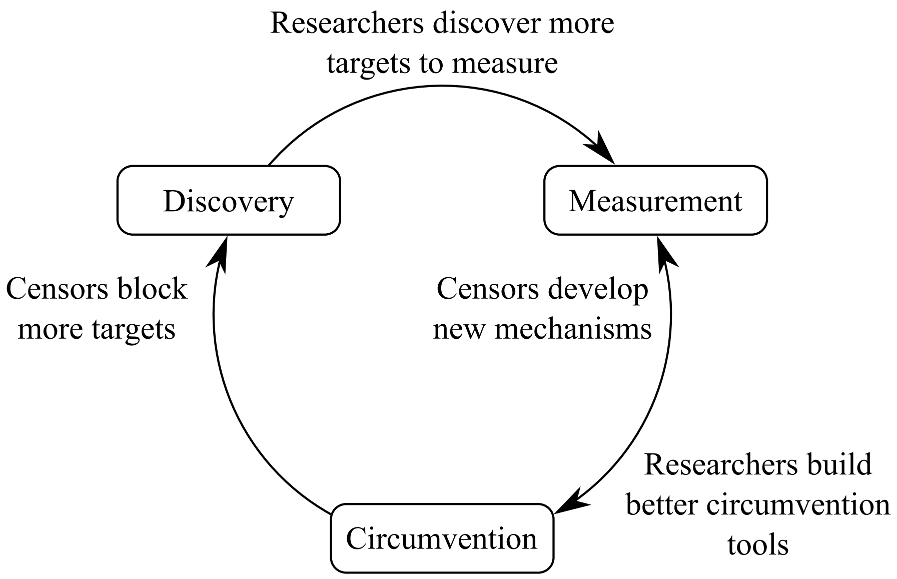

Researchers discover more targets to measure
Discovery
Measurement
Censors block more targets
Censors develop new mechanisms
Researchers build better circumvention tools
Circumvention

Figure 3: Three phases of Internet censorship each influence each other. Censors and researchers wage an arms race against each other.

circumvention techniques we introduce in this chapter (or the rest of this dissertation) will be relevant very far in the future.

2.3 Discovery

Discovery aims to produce a list of targets to test for censorship, *e.g.*, a list of potentially censored URLs. This typically relies on the knowledge of regional experts (*e.g.*, activists, reporters, government insiders, or users in affected countries who happen to experience censorship firsthand). There are several sources for such lists. Policy organizations publish findings of censorship practices around the world. For example, the Open Network Initiative has published qualitative reports based on measurements from a limited number of vantage points, with scant insight into how censorship evolves over short timescales or what exactly is being filtered [112,113]. Other projects such as Herdict [82], GreatFire [77], and Filbaan [65] maintain lists of domains that may be blocked, usually contributed by volunteers. Herdict compiles reports from users about domains that are blocked from a certain location; such reports lack independent verification. GreatFire monitors reachability of various domains and services from a site behind China's censorship firewall; it also maintains historical measurement results. Each of these tools offers only limited information that is driven by user-initiated reporting or measurements, yet these services and reports can serve as initial lists of URLs to test for censorship with more systematic approaches.

While most target lists ultimately rely on human knowledge, some projects attempt to discover censorship with no *a priori* information about what might be censored by inferring possible censorship events from service usage logs. The Tor Censorship Detector attempts to infer blocking of Tor by detecting abnormal decreases in Tor traffic [44, 141]. Similarly, Kathuria [94] attempted to detect censorship of a single service, the BBC [13], from its per-country usage statistics. Thus far, these techniques are plagued by false positives and are generally too noisy to be useful.

2.4 Measurement

After discovering a list of potential censorship targets, we aim to collect measurements that confirm or refute the existence of censorship by telling us what targets are censored, where they are censored, when they are censored, and how censors implement the censorship (*i.e.*, the mechanisms they use).

Table 2 summarizes prior measurement studies of Internet censorship. Rather than introduce every study in painstaking detail, we explain the kinds of measurements of network censorship that these studies collect via a taxonomy of characteristic features of any measurement collection scheme. This taxonomy should also be helpful when interpreting the results of future studies.

Vantage points. A vantage point is some node on the network which collects censorship measurements. It could be a laptop in a user's home, a smartphone, dedicated measurement collection infrastructure like a RIPE Atlas Probe [126], a Virtual Private Server (VPS), a PlanetLab node [12], a SOCKS proxy, a VPN endpoint, or occasionally even a censorship middlebox itself [3]. Because vantage points determine the representativeness of conclusions we can draw and the breadth of experiments we may run, selection of vantage points is often a measurement study's most defining characteristic. Implementation of censorship may vary widely between ISPs and coexist with localized (non-censorship) network failure, so studies often strive to collect data from a redundant set of vantage points from multiple ASes in a country.

Most measurement studies aim to characterize the censorship seen by *real users* in a country (*i.e.*, not just users in educational networks or data centers). Although placing collection software or hardware in the hands of real users is tempting because of the quality of data they can collect, researchers must (1) find users in these countries willing to host such measurement software or hardware and (2) ensure that running these measurements doesn't place users in danger (*e.g.*, collecting such measurements in oppressive regimes with strict censorship controls could be illegal).

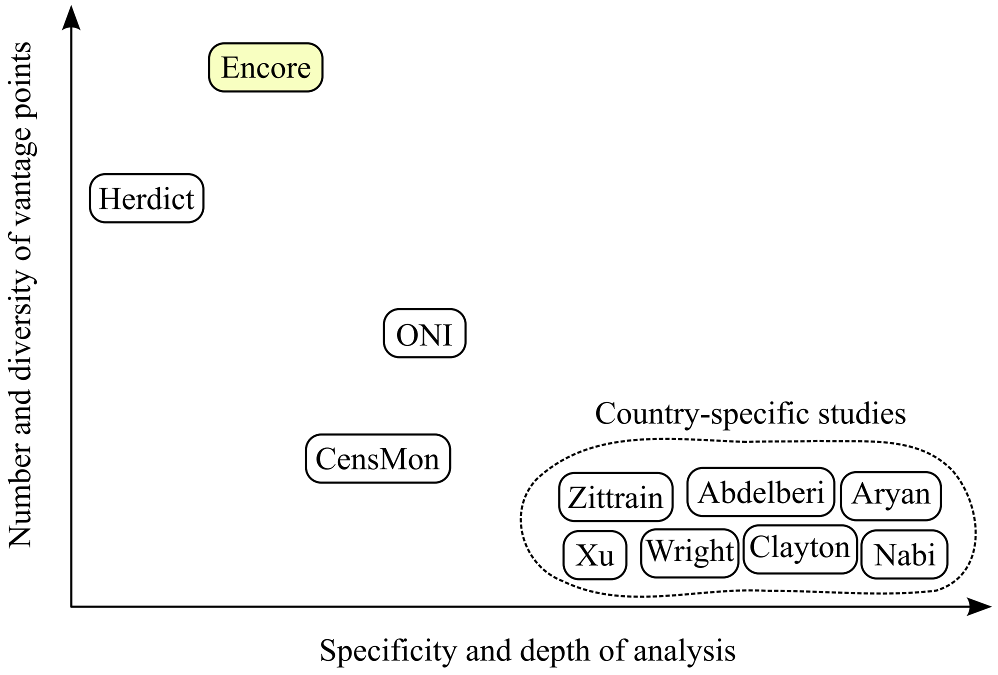


Figure 4: Comparison of censorship measurement studies by breadth and depth. Many studies analyze a single country in great depth, while a few attempt to compare censorship across countries and regions. In Chapter 3, we introduce Encore, which enables massive deployment of Web censorship measurements. Refer to Table 2 for study descriptions.

Table 2: Existing censorship measurement collection systems and studies, categorized by geographic scope and ordered reverse chronologically. A *client* vantage point is one where measurements are collected on a consumer laptop/desktop in a residential or educational setting. *China keywords* refers to a small, well-known set of keywords that trigger the Chinese firewall (*e.g.*, perhaps the most famous is "falun," denoting the Falun Gong movement suppressed by the Chinese government). *Web* indicates a set of common Web censorship mechanisms.

Name	Vantage points	Scope	Duration	Target list	Mechanisms
		Multiple countries			
Herdict [82]	Clients (Web browsers)	245 countries	5 years, 2009 – 2014	User generated, regional experts	Web
ONI [47, 48, 74]	Clients in 286 ISPs	77 countries	5 years, 2007 – 2012	Regional experts	Web
Verkamp *et al.* [153]	PlanetLab, proxies, clients	11 countries	Sometime in 2012	Herdict [82]	Web
CensMon [135]	PlanetLab	33 countries	14 days in 2011	User generated, Google Alerts, Twitter, Herdict, ONI	Web
Dainotti [41]	Various [28, 29, 98, 110, 128]	Egypt, Libya	4 months in 2011	N/A (passive)	AS
Net Neutrality Monitor [107]	Unknown	7 countries	22 (unknown)	User generated	IP, DNS blocking
		Country-specific			
Nabi *et al.* [105]	5 clients in 5 ISPs	Pakistan	2 months in 2013	[17]	Web, keyword
Aryan *et al.* [6]	1 client	Iran	2 months in 2013	Alexa 500 [4]	Web, keywords, throlling
Winter *et al.* [159]	32 SOCKS Proxies, 1 VPS	China	3 weeks in 2012	N/A (service)	See paper [159]
Wright [161]	Clients in UK	China	1 day in 2012	Herdict [82]	DNS
Abdelberi [3]	7 censorship middleboxes	Syria	2 months in 2011	N/A (passive)	Web, keywords
Xu *et al.* [165]	PlanetLab outside China	China	2 months in 2010	Regional sites	Keyword
Park *et al.* [119]	Proxies	China	3 months, 2008 – 2009	China keywords	TCP RST from HTML response
ConceptDoppler [39]	Clients in US	China	Weeks in 2008	Keywords from Wikipedia [39]	TCP RST
Clayton *et al.* [35]	Clients in UK	China	10 days in 2006	China keywords	TCP RST
Zittrain *et al.* [168]	Modems, proxies	China	5 months in 2002	Search results	DNS injection, IP filtering, keyword

Because of these difficulties, many collection efforts use vantage points that only approximate the conditions seen by real clients. For example, studies relying PlanetLab, VPSes, or proxies often come with caveats that their results might not be representative of "real" users if censorship policies differ across ISPs. Several studies of the Chinese firewall rely on the symmetric behavior of its keyword filtering: instead of using a vantage point with China to send traffic, they send traffic from vantage points outside China to arbitrary hosts inside China that nonetheless trigger keyword filtering [35, 39, 161]. Figure 4 illustrates the trade off between breadth and depth that measurement studies typically make.

Geographic focus. Prior measurement efforts typically fall into two classes. The first studies a common censorship mechanism across a variety of countries that employ that mechanism; the second studies the nuances of censorship in a single country. Comparing results collected from different countries may become easier as more countries adopt commercial censorship equipment [42].

Collection duration. With only a few exceptions, prior measurement efforts only last a few weeks or months. Unfortunately, governments often change their censorship policies, either to suppress circumvention efforts or in response to extraneous political events (*e.g.*, elections), which limits the conclusions we may draw from short studies; long-term collection is often the only way to fully capture such phenomena.

Consistently collecting data during the study is equally important. A study that takes only a few measurements at irregular intervals often doesn't provide granular enough information to rule out the possibility of spurious network errors (*e.g.*, unreliable links) rather than targeted censorship.

Source of target lists. As we mentioned in Section 2.3, researchers curate target lists using a variety of techniques. Some studies rely on lists compiled by third parties (*e.g.*, organizations like the ONI or Herdict), while others compile their own test lists using custom methods (*e.g.*, crawling search engine results, examining Twitter trends, etc.). Fragmentation of target lists is problematic; it is difficult to compare results between studies using different target lists, even within the same country.

Some studies don't require a target list because they analyze passively collected data or only consider censorship of entire services (*e.g.*, Tor, TLS) [3, 159].

Censorship mechanisms. Most countries focus on Web censorship; that is, they disrupt some stage of an HTTP or HTTPS connection, as we described in Figure 2. The breadth of studies of Web censorship reflects this. Other studies focus on techniques commonly used in China (keyword filtering, TCP reset injection, and DNS spoofing) or less granular mechanisms like AS-level blocking.

Table 3: Existing network measurement collection platforms that could support future studies of Internet censorship. Some platforms would require modification before they could collect censorship measurements.

Name	Vantage points	Flexibility
No modification required		
Dasu [131]	Hundreds of thousands	ping, traceroute, HTTP GET, DNS
Seattle [30, 132]	Tens of thousands	Restricted Python [125]
RIPE Atlas Probe [126]	Thousands	ping, traceroute, DNS, verify SSL certs [127]
PlanetLab [12, 120, 122]	Hundreds (universities)	Linux
BISmark [16, 138]	Hundreds (homes)	OpenWrt [115]
Fathom [49]	Hundreds?	Fathom API [58]
OONI [64]	Dozens	Python
Modification required		
Netalyzr [96]	Hundreds of thousands	Java
MySpeedTest [104]	Hundreds (mobile)	Android

2.4.1 Potential measurement platforms

Given the importance of choosing good vantage points, future research efforts could focus on bringing censorship measurements to existing measurement platforms. Table 3 summarizes a few existing measurement platforms. The feasibility of adopting a particular platform for censorship measurement depends on the number of vantage points on which it currently runs, the diversity of those vantage points, the flexibility of measurement tasks that it could support, and whether we would need to modify its existing source code (rather than running new measurement collection programs using the platform's existing control interfaces). Such an effort may also grapple with challenges of trust when attempting to use a platform not for its advertised purpose.

At the time of writing, we are aware of active efforts to deploy censorship measurements using Seattle, BISmark, OONI, and MySpeedTest.

2.5 Censorship circumvention

We stated in Section 2.1 that Internet censorship is often just application of standard network security techniques in new contexts. Likewise, censorship circumvention plays the role of the adversary in traditional network security by trying to find weaknesses in these security mechanisms. Studying circumvention can benefit both those affected by censorship (*i.e.*, by improving circumvention technologies) and developers and researchers seeking to find and fix weaknesses in security solutions.

To communicate with a censorship circumvention tool, a user must typically follow three steps:

1. The user must learn who to communicate with. Users often circumvent address-based censorship by communicating via a third party, but before doing so must learn who that third party is. This isn't always a hard problem; for example, in countries that only practice content-based censorship, it isn't necessary to communicate via a third party and users can send data directly to its destination, so users often already know who to communicate with. In countries that practice address-based censorship, users typically need to learn the address of some proxy server or service that can rely traffic to its final destination.
2. Next, the user must learn how to communicate. Sometimes this is trivial, *e.g.*, simply forward traffic to an open proxy server, which will relay the traffic to its destination. In many cases, this step involves distribution of encryption keys and software that uses those keys to establish secure communication.
3. Using these two pieces of information, the user can establish communication using the circumvention tool.

The first two steps are called *bootstrapping*. Because the exchange of addresses and credentials during bootstrapping is an activity that itself may be subject to censorship, users may need to compose multiple circumvention tools by using a slow, secure secure circumvention tool to exchange bootstrapping information for a faster tool.

Developers of censorship circumvention systems often strive for several goals:

- *Resiliency.* The circumvention system must enable users to successfully evade some type of censorship.
- *Performance.* Communication should be as fast as possible while still being secure. We typically measure performance of circumvention systems using traditional network performance metrics like latency and throughput. There is a general tradeoff between performance and other goals; systems that achieve anonymity and deniability typically have far lower performance.
- *Deployability.* Circumvention tools must make it into the hands of real users and must be usable by those users. Systems strive to reduce their bootstrapping cost, *i.e.*, the cost of obtaining initial software and keys, which must happen necessarily happen outside of the circumvention tool itself.
- *Confidentiality.* The censor should be unable to read the contents of communication. This property is normally provided by traditional encryption and is useful because it

can shield application traffic, thereby frustrating attempts to censor based on traffic content. For example, a censor wishing to block a single HTTPS URL will have trouble doing so without inducing collateral damage.

- *Anonymity.* The censor should be unable to discover who a user is communicating with. Anonymity normally refers to anonymity of IP addresses, but also applies to anonymity of users or services. This property is useful because censors often try to restrict who users communicate with (as opposed to the contents of their communication). Although useful for censorship circumvention, people also use anonymous communication tools for other reasons. Tor [51] is probably the most successful anonymous communication tool and has a many users in countries that do not censor Internet access.
- *Deniability.* The censor should be unable to discover that the user is attempting to communicate using a circumvention tool. This property is useful because making it difficult for a censor to identify users of circumvention tools increases the amount collateral damage induced by blocking such tools. For example, if a tool generates traffic that resembles a VoIP phone call, then a censor wishing to block that tool may need to block all VoIP communications; doing so would impose a much greater cost on the censor.

We now summarize approaches and systems for censorship circumvention and discuss which of the goals they meet. Remember that users may compose multiple tools to enhance their security; in fact some tools are designed explicitly for this purpose by, *e.g.*, specifying only a bootstrapping technique and relying on other tools for actual communication.

Simple proxies. Probably the most widely used circumvention tools are simple proxy tools, like VPNs, SSH tunnels, SOCKS proxies, and HTTP proxies. Standard client and server software (*e.g.*, OpenVPN, OpenSSH, etc.) often works, although developers have also created specialized software that adds features to improve resiliency, like automatic rotation through a pool of potentially blocked proxy IP addresses [70, 123, 151]. Proxies are attractive because they usually have high performance (*i.e.*, they route traffic through a single relay before sending it to its destintaion) and are very easy to deploy once users learn their addresses because proxy functionality is built into most operating systems. These services do not prescribe a method for bootstraping: users may obtain proxy addresses by word-of-mouth, Web sites that provide lists of addresses, purchasing access from an overseas VPN provider, etc. This is one weakness of these services; most methods for learning proxy addresses make them either easy to find and therefore easy for the censor to block, or hard for the censor to block and therefore hard to find. Additionally, proxies are neither anonymous, confidential, nor deniable with respect to the proxy operator, and require that users trust this operator, who could collude with authorities to divulge users' activities. uProxy [152] and Kaleidoscope [137] enable users to relay their traffic via their social network to alleviate the risk of trusting proxies.

Anonymizing mix networks. Mix networks (*e.g.*, Tor [51], Tarzan [67], Mixminion [46], Crowds [124]) improve on simple proxies by offering a network of machines through which users can send traffic if they wish to communicate anonymously with one another. Danezis and Dias present a comprehensive survey of these networks [45]. Compared to simple proxies, mix networks trade performance for anonymity and confidentiality, although previous work has shown that, depending on its location, a censor or observer might be able to link sender and receiver [7, 11, 61, 103, 133, 134]. These systems do not provide deniability, but users can easily compose mix networks with covert communication channels as we describe next. Finally, mix networks like Tor traditionally use a public relay list which is easily blocked, although work has been done to try to rectify this [140, 142].

Covert channels. To achieve deniability, several projects have designed covert communication channels that hide their traffic inside seemingly legitimate cover. Many of these systems are designed to be composed with another circumvention tool to provide additional security (*e.g.*, anonymity), most as *pluggable transports* for Tor [50]; there are pluggable transports that disguise Tor traffic as VoIP calls [86, 102], HTTP [157], encrypted traffic [109], peer-to-peer communications [62], or arbitrary protocols [53, 160]. Other schemes like Infranet [59] operate independently of Tor. Researchers have identified weaknesses in common approaches to building these transports [73, 85]; in Chapter 4 we present Collage, which provides a covert channel that avoids some of these concerns.

Rendezvous protocols. Some covert channels are too slow for interactive use and are used as rendezvous protocols for exchanging small pieces of information, particularly addresses and keys for faster circumvention tools. They typically achieve very strong security properties like anonymity and deniability at the cost of performance. These sytems include general purpose rendezvous tools like Online Scanning Services [63] and Collage (Chapter 4); and systems designed specifically to exchange a particular kind of bootstrapping material, like proxy addresses [60, 101, 137], or Tor relay node addresses [97, 155].

In-network relays. Several researchers have proposed systems that move relays into the middle of the network [85, 93, 163]. Users address flows to unsuspicious and unwitting destinations (*e.g.*, weather.com) and a router in the middle of the Internet redirects them to another, censored destination. This method provides both anonymity and deniability from the censor, but is very unlikely to be deployed because it requires modification of core

Internet infrastructure.

Censorship-resistant publication. Some systems aim to provide censorship-resistant document storage (as opposed to simply acting as a relay to services that host documents). Some systems allow publishers and clients to exchange content using either peer-to-peer networks (Freenet [34]) or using a storage system that makes it difficult for an attacker to censor content without also removing legitimate content from the system (Tangler [154]). Freenet provides anonymity and unlinkability, but does not provide deniability for users of the system, nor does it provide any inherent mechanisms for resilience: an attacker can observe the messages being exchanged and disrupt them in transit. CovertFS [9] is a file system that hides data in photos using steganography; it is similar to Collage, but provides fewer security guarantees.

Neat circumvention tricks. Several researchers have identified weaknesses in specific censorship implementations that enable circumvention with little to no performance overhead. Clayton *et al.* [35] discovered that hosts configured to ignore TCP resets can circumvent the Chinese firewall, while Duan *et al.* [52] describe how to ignore injected DNS responses. While fast, these techniques are brittle because they rely on features of specific implementations; they also provide no security.

2.6 How this chapter fits with the rest of the dissertation

Chapter 3 presents a method for measuring several varieties of Web censorship from a broad spectrum of countries. Chapter 4 presents a new rendezvous protocol that provides deniability and confidentiality and is resilient against a far greater range of attackers than most existing circumvention techniques.

CHAPTER III

MEASURING WEB CENSORSHIP WITH CROSS-ORIGIN REQUESTS

3.1 Introduction

Consistently and reliably gathering measurements of Internet censorship is extremely difficult. As we discussed in Chapter 2, perhaps the biggest obstacle entails obtaining access to a diverse and globally distributed set of vantage points, particularly in regions that are most likely to experience Internet censorship. Achieving such a widespread deployment in these locations often requires surmounting language and cultural barriers and convincing users to install measurement software that is specifically designed for measuring censorship. In some countries, the operation or even possession of such specialized measurement software could itself be risky or even illegal. Researchers have begun to develop custom tools to measure filtering from a diverse set of clients worldwide (*e.g.*, OONI [64, 114], Censorscope [31]), yet widespread deployment remains a perennial challenge. As a result, researchers have resorted to informal data collection (*e.g.*, complaints on forums [143]) or collection from a small number of non-representative vantage points (*e.g.*, PlanetLab nodes [135], hosts on virtual private networks, or even ephemeral one-off deployments of single vantage points) that might not observe the filtering policies that many mobile and home users experience.

We take an alternate approach: rather than asking users to deploy custom censorship measurement software—which can be both cumbersome and potentially incriminating—we take advantage of existing features of the Web to induce unmodified browsers to perform measurements of Web censorship. Most users access the Internet using a Web browser, so if we can induce these browsers to perform censorship measurements, we can collect data from a much larger, more diverse, and more representative set of vantage points than is possible with custom censorship measurement tools.

Our system, Encore, uses Web browsers on nearly every Internet-connected device as potential vantage points for collecting data about what, where, and when Web filtering occurs. Encore relies on a bystander who is hosting a Web page to include a one-line embedded script, which attempts to retrieve content from third-party Web sites using cross-origin requests [22, 130]. Although same-origin policies in browsers prohibit many such requests (*e.g.*, to thwart cross-site request forgery), our work demonstrates that the types of cross-origin requests that browsers *do* allow are sufficient to collect information and draw conclusions about Web filtering. The Encore script induces every visitor of this Web page to request an object from a URL that Encore wishes to test for filtering. Encore then sends binary feedback about whether that request succeeded or failed to a central server for correlation and analysis across vantage points.

Although the basic mechanism is conceptually simple, designing Encore presents several challenges. First, Encore's measurements must operate within the constraints of the types of cross-origin requests that Web browsers permit. For example, the `img` HTML directive yields the most conclusive feedback about whether an object fails to load, but can only be used to test images, not general URLs. This limitation means that while it may be useful for detecting (say) the filtering of an entire DNS domain, it cannot test the reachability of specific (non-image) URLs. Encore's design must recognize which cross-origin requests browsers permit and use collections of these requests to draw inferences with higher confidence. Second, because Encore requires operators of Web sites to augment their existing Web pages, Encore should be easy to install and incur minimal performance overhead on the Web sites where it is deployed.

Encore can detect *whether* certain URLs are filtered, but it cannot determine *how* they are filtered. Subtler forms of filtering (*e.g.*, slowing page loads by introducing latency or packet loss) are more difficult to detect, and detecting manipulation of the content itself (*e.g.*, replacing a Web page with a block page, or substituting content) using Encore is nearly impossible. As such, Encore may ultimately complement other censorship measurement systems, which can perform detailed analysis but face much higher deployment hurdles. Ultimately, neither Encore nor other censorship analysis tools can determine human motivations behind filtering, or even whether filtering was intentional; they only provide data to policy experts who make such judgments.

We present background on cross-origin Web requests in Section 3.2. Section 3.3 explains how Encore infers Web filtering using cross-origin requests, and Section 3.4 explains how to apply this inference technique and gather measurements from a diverse population of clients. We evaluate the feasibility of deploying Encore in Section 3.5, explore preliminary results in Section 3.6, and conclude with a discussion of security in Section 3.7.

3.2 Background

This section gives an overview of cross-origin requests, which are the technique that Encore uses to measure Web filtering. We then introduce threats to our accurate collection and interpretation of these measurements.

3.2.1 Cross-Origin Requests

Web browsers' *same-origin policies* restrict how a Web page from one origin can interact with a resource from another origin; an *origin* is defined according to the protocol, port, and DNS domain ("host") [130]. Sites can typically send information to another origin using links, redirects, and form submissions, but they cannot generally receive data from another; in particular, browsers restrict cross-origin reads from scripts to prevent attacks such as cross-site request forgery. However, because cross-origin *embedding* is typically allowed, certain read access can be leaked by means of embedding. The cornerstone of Encore's design is to use information that is leaked via cross-origin embedding of objects to determine whether an object from another origin can be successfully loaded.

Various mechanisms allow Web pages to embed remote resources using HTTP requests across origins; some forms of cross-origin embedding are not subject to the same types of security checks as other cross-origin reads. Examples of resources that can be embedded include simple markup of images or other media (*e.g.*, `<img>`), remote scripts (*e.g.*, `<script>`), remote stylesheets (*e.g.*, `<link rel="stylesheet" href="...">`), embedded objects and applets (*e.g.*, `<embed src="...">`), and document embedded frames such as iframes (*e.g.*, `<iframe>`). Each of these remote resources has different restrictions on how the origin page can load them, and the amount of information that is leaked from such embedded references. For example, images embedded with the `img` tag trigger an `onload` event if the browser successfully retrieves and renders the image, and an `onerror` event otherwise. The ability for the origin page to see these types of events can allow the origin page to infer whether the cross-origin request succeeded, even though the request refers to an object from a different origin.

Although cross-origin embedding of media provides the most explicit feedback to the origin about whether the page load succeeded, other embedded references can still provide more limited information, through timing of `onload` invocation or introspection on a Web page's style. Additionally, different browsers have different security policies and vulnerabilities; for example, we discovered that the Chrome browser allows an origin site to load any cross-origin object via the `script` tag, which allows us to perform a much more liberal set of measurements from Chrome users. One challenge in designing Encore is determining whether (and how) various embedded object references can help infer information about whether an object was retrieved successfully.

3.2.2 Prior use of cross-origin requests

We are not the first to advocate the use of cross-origin requests for client measurements. Bortz *et al.* use timing information from cross-site requests to infer various information, such as whether a user is logged into a particular site or whether a user has visited a Web page before [19]. Karir *et al.* use embedded Javascript with cross-origin requests to measure IPv6 reachability and performance from large numbers of clients [92]; the use of embedded cross-origin requests to obtain large number of clients is similar to Encore's design. Other systems have used cross-origin requests to third parties to determine information such as network latency between a client and some other Internet destination [79, 108]. These tools and techniques use similar techniques as Encore, but they primarily aim to measure network performance or past user behavior based on the timing of successful cross-origin requests. They do not include techniques to infer reachability of domains, IP addresses, or URLs based on the success (or lack thereof) of cross-origin requests. This section surveys related work. We summarize existing censorship measurement techniques and previous studies of Internet censorship and Web filtering; other policy reports of Internet censorship (which can ultimately seed our measurements); and other efforts to perform measurements from clients using advertisements or embedded images. Although we broadly survey Internet censorship practices, Encore focuses specifically on Web filtering, not more general Internet censorship.

3.2.3 Threat Model

We assume the censor applies the Web filtering mechanisms we introduced in Section 2.1. Because of censors implement these mechanisms in various ways, we conservatively assume that censors can impose censorship from any network infrastructure within their control, excluding end hosts. (Previous attempts at widespread control over end hosts have been unsuccessful [57].) Web filtering typically takes place when the client performs an initial DNS lookup (at which point the DNS request may result in blocking or redirection), when the client attempts to establish a TCP connection to the Web server hosting the content (at which point packets may be dropped or the connection may be reset), or in response to a specific HTTP request or response (at which point the censor may reset the TCP connection, drop HTTP requests, or redirect the client to a block page). Encore can usually determine when a page is blocked, but in most cases cannot determine how. Section 3.3 details Encore's inference algorithm for detecting various forms of Web filtering using cross-origin requests.

Our goal is not to circumvent Web filtering but to measure it. Nevertheless, measurement faces its own set of threats. Suppose that clients measure Web filtering by attempting to reach a site and subsequently submitting reports on whether those attempts succeeded or failed. In such a scenario, we assume an adversary that can reject, block, or modify any stage of a Web connection in order to filter Web access for subsets of clients, although we assume the adversary uses a blacklist and is unwilling to filter all Web traffic, or even significant fractions of all Web traffic. This adversary plays three roles in Encore's design: (1) the goal of Encore is to measure the adversary's Web filtering behavior, (2) the adversary may attempt to filter clients' access to Encore itself, thereby preventing them from collecting or contributing measurements, and (3) the adversary may attempt to distort Encore's filtering measurements by allowing measurement traffic but denying "regular" access to the same site. We consider all three aspects of Encore's relationship with this adversary. Additionally, we briefly consider an adversary that attempts to block, corrupt, or falsify either the measurement tasks we distribute to clients or the measurement results we collect from them.

3.3 Measuring Filtering with Encore

This section explains how Encore measures Web filtering using cross-origin requests. We first provide an overview of how Encore works, and describe the specific limitations on what Encore can and cannot conclude from the information it collects. We then describe the measurement tasks that Encore can perform, and what it can (and cannot) infer from these measurements. Finally, we describe the setup that we use to validate Encore's measurements. Section 3.4 builds a distributed Web filtering measurement platform using the measurement and inference primitives that we develop in this section. We defer discussion of how origin Web servers learn which measurement targets to test from each client (*target selection*) and how Web clients send the results of their measurements to a central location for analysis (*measurement collection*) to Section 3.4.

3.3.1 Overview

Figure 5 illustrates how Encore measures Web filtering. The process involves three parties: a Web client that acts as our measurement vantage point; a measurement target, which is a server that hosts a Web resource that we suspect is filtered; and an origin Web server, which serves a Web page to the Web client instructing the client how to collect measurements. In each page it serves, the origin includes a *measurement task*, which is a small measurement collection program that attempt to access potentially filtered Web resources

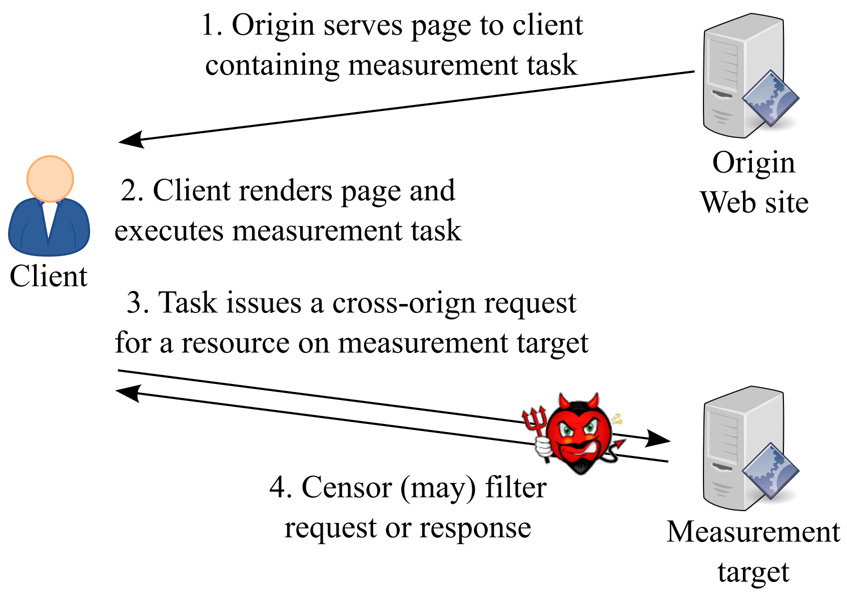

1. Origin serves page to client containing measurement task
Origin Web site
2. Client renders page and executes measurement task
Client
3. Task issues a cross-orign request for a resource on measurement target
4. Censor (may) filter request or response
Measurement target

Figure 5: Encore induces browsers to collect Web filtering measurements by bundling measurement tasks inside pages served by an origin Web site. Measurement tasks instruct the client to attempt to fetch a Web resource from a measurement target using a cross-origin request.

(*e.g.*, Web pages, image files, etc.) from the target and determine whether such accesses were successful. The client runs this measurement task after downloading and rendering the page. The greatest challenge is coping with browsers' limited APIs for conducting network measurements, particularly when accessing these resources requires issuing cross-origin requests.

The scope of Web filtering varies in granularity from individual URLs (*e.g.*, a specific news article or blog post) to entire domains. Detecting Web filtering is difficult regardless of granularity. On one hand, detection becomes more difficult with increasing specificity. When specific Web resources are filtered (as opposed to, say, entire domains), there are fewer ways to detect it. Detecting filtering of entire domains is relatively straightforward because we have the flexibility to test for such filtering by simply checking accessibility of a small number of resources hosted on that domain. In contrast, detecting filtering of a single URL essentially requires an attempt to access that exact URL. Resource embedding only works with some types of resources, which further restricts the Web resources we can test and exacerbates the difficulty of detecting very specific instances of filtering.

On the other hand, inferring broad filtering is difficult because Encore can only observe the accessibility of individual Web resources, and such observations are binary (*i.e.*, whether or not the resource was reachable). Any conclusions we draw about the scope of

Web filtering must be inferred from measurements of individual resources. We take a first-order glimpse at such inferences in Section 3.3.3 and present a filtering detection algorithm in Section 3.6.

3.3.2 Measurement tasks

Measurement tasks are small, self-contained HTML and JavaScript snippets that attempt to load a Web resource from a measurement target. Encore's measurement tasks must satisfy four requirements:

1. They must be able to successfully load a cross-origin resource in locations without Web filtering. This excludes using XMLHttpRequest (*i.e.*, AJAX requests), which is the most convenient way to issue cross-origin requests, because default Cross-origin Resource Sharing settings prevent such requests from loading cross-origin resources from nearly all domains. Instead, we induce cross-origin requests by embedding images, style sheets, and scripts across domains, which browsers typically allow.

2. They must provide feedback about whether or not loading a cross-origin resource was successful. Several convenient mechanisms for loading arbitrary cross-origin requests (*e.g.*, the `iframe` tag) lack a clear way to detect when resources fail to load, and are hence unsuitable for measurement tasks.

3. Tasks must not compromise the security of the page running the task. Tasks face both client- and server-side security threats. On the client side, because Encore detects Web filtering by *embedding* content from other origins (rather than simply requesting it, as would be possible with an AJAX request), such embedding can pose a threat as the browser renders or otherwise evaluates the resource after downloading it. In some cases, rendering or evaluating the resources is always innocuous (*e.g.*, image files); in other cases (*e.g.*, JavaScripts), Encore must carefully sandbox the embedded content to prevent it from affecting other aspects of Web browsing. Requesting almost any Web object changes server state, and measurement tasks must take these possible side effects into account. In some cases, the server simply logs that the request happened, but in others, the server might insert rows into a database, mutate cookies, change a user's account settings, etc. Although it is often impossible to predict such state changes, measurement tasks should try to only test URLs without obvious server side-effects.

4. Finally, measurement tasks must not significantly affect perceived page performance, appearance, or network usage.

Table 4: Measurement tasks use several mechanisms to discover whether Web resources are filtered. Each mechanism can only be used to test for filtering of a subset of Web resources. Some resources cannot be tested using any of these methods. We empirically evaluate parameters for images and inline frames in Section 3.5.

Embedded resource	Summary	Limitations
Images	Render an image. Browser fires `onload` if successful.	Only small images (*e.g.*, $\leq$ 1 KB).
Style sheets	Load a style sheet and test its effects.	Only non-empty style sheets.
Inline frames	Load a Web page in an iframe, then load an image embedded on that page. Cached images render quickly, implying the page was not filtered.	Only pages with cacheable images. Only small pages (*e.g.*, $\leq$ 100 KB). Only pages without side effects.
Scripts	Load and evaluate a resource as a script. Chrome fires `onload` iff it fetched the resource with HTTP 200 status.	Only with Chrome. Only with strict MIME type checking.

Below is an example of a simple measurement task that instructs the Web client to load an image hosted by a measurement target `censored.com`:

```
<img src="//censored.com/favicon.ico"
     style="display: none"
     onload="submitSuccess()"
     onerror="submitFailure()"/>
```

This task meets the four requirements because it (1) requests an image from a remote measurement target using the `img` tag, which is allowed by browser security policy, (2) detects whether the browser successfully loaded the image by listening for the `onload` and `onerror` events, (3) trivially maintains security by not executing any code from resources served by the measurement target, and (4) preserves performance and appearance by only loading a very small icon (typically 16x16 pixels) and hiding it using the `display: none` style rule. Section 3.4 explains how tasks submit results using the `submitSuccess` and `submitFailure` functions. Appendix A presents a longer example of a measurement task.

3.3.3 Inferring Web filtering

A measurement task provides binary indication of whether a particular resource failed to load, thus implying filtering of that specific resource. From this, we draw more general conclusions about the scope of filtering, beyond individual resources (*e.g.*, whether an entire domain is filtered, whether an entire portion of a Web site is filtered, whether certain keywords are filtered). We must do so with little additional information about the filtering mechanism. This section describes how we design sets of measurement tasks to make these inferences in the abstract. We defer details about how to use these tasks to measure specific instances of filtering Section 3.4.

There are several ways of testing accessibility of cross-origin Web resources; unfortunately, none of them work across all types of filtering, all Web browsers, and all target sites. Instead, we automatically tailor measurement tasks to each measurement target and Web client. Detecting Web filtering gets harder as the scope of filtering becomes more specific, so we start with techniques for detecting broad-scale filtering and work toward more specific filtering schemes. Table 4 summarizes the measurement tasks we discuss in this section. Section 3.5 empirically determines concrete values for vague claims about requirements for each measurement task (*e.g.*, “small” images).

3.3.3.1 Filtering of entire domains

Encore can perform collections of measurement tasks that can help us infer that a censor is filtering an entire domain. It is prohibitively expensive to check accessibility of *every* URL hosted on a given domain. Instead, we assume that if several “auxiliary” resources hosted on a domain (*e.g.*, images, style sheets, etc.) are inaccessible then the entire domain is probably inaccessible. Our intuition is that the adversary is unlikely to filter access to merely an arbitrary sampling of supporting resources (as there is not really a practical reason for doing so); rather, the adversary will more likely filter the entire domain (or at least an entire section of a Web site).

Fortunately, detecting filtering of some auxiliary resources is straightforward because these resources are typically embedded on Web pages even across origins. We discuss two such auxiliary resources.

Images. Web pages commonly embed images, even across origins. Such embedding is essential for enabling Web services like online advertising and content distribution networks to serve content across many domains.

Encore attempts to load and display an image file from a remote origin by embedding it using the `<img src=...` tag. Conveniently, all major browsers fire an `onload` event after the browser fetches and renders the image, and fire `onerror` if either of those steps fails; the requirement to successfully render the image means that this mechanism only works for image files and cannot decide the accessibility of non-image content. Downloading and rendering an image does not affect user-perceived performance if the image is small (*e.g.*, an icon), and measurement tasks can easily hide images from view. This

technique only works if the remote origin hosts a small image for us to embed.

Style sheets. Web pages also commonly load style sheets across origins. For example, sites often load common style sheets (*e.g.*, Bootstrap [18]) from a CDN to boost performance and increase cache efficiency.

Encore attempts to load a style sheet using the `<style src=...>` tag and detects success of the load by verifying that the browser applied the style specified by the sheet. For example, if the sheet specifies that the font color for `<p>` tags is blue, then the task creates a `<p>` tag and checks whether its color is blue using `getComputedStyle`. To prevent the sheet's style rules from colliding with those of the parent Web page, we load the sheet inside an `iframe`. Although some browsers are vulnerable to cross-site scripting attacks when loading style sheets, these issues have been fixed in all newer browsers [88]. Style sheets are generally small and load quickly, resulting in negligible performance overhead.

3.3.3.2 Filtering of specific Web pages

Governments sometimes filter one or two Web pages (*e.g.*, blog posts) but leave the remainder of a domain intact, including resources embedded by the filtered pages [105]. Detecting this type of filtering is more difficult because there is less flexibility in the set of resources that Encore can use for measurement tasks: it must test accessibility of the Web page in question and cannot generally determine whether the page is filtered based on the accessibility of other (possibly related) resources. Testing filtering of Web pages, as opposed to individual resources, is significantly more expensive, complicated, and prone to security vulnerabilities because such testing often involves fetching not only the page itself, but also fetching all of that page's referenced objects and rendering everything. This means we must be very careful in selecting pages to test. Many pages are simply too expensive or open too many vulnerabilities to test. Section 3.4 discusses the infrastructure and decision process we use to decide whether a Web page is suitable for testing.

We present two mechanisms for testing Web filtering of Web pages, and the limitations of each mechanism:

Inline frames. A Web page can include any other Web page inside itself using the `iframe` tag, even across origins. However, browsers place strict communication barriers between the inline page and the embedding page for security, and provide no explicit notification about whether an inline frame loaded successfully.

Instead, the task infers whether the resource loaded successfully by observing timing. It first attempts to load the page in an iframe; then, after that iframe finishes load,the task records how long it takes to download and render an image that was embedded on that page. If rendering this image is fast (*e.g.*, less than a few milliseconds) we assume that the image was cached from the previous fetch and therefore the Web page loaded successfully. This approach obviously only works with pages than embed objects that will be cached by the browser and are unlikely to have been cached from a prior visit to another Web page; for example, common images like the Facebook's "thumbs up" icon appear on many pages and may be in the browser cache even if the iframe failed to load. This approach can be expensive because it requires downloading and rendering entire Web pages. Additionally, pages can detect when they are rendered in an inline frame and may block such embedding.

Scripts. Web pages often embed scripts across origins, similarly to how they embed style sheets. For example, many pages embed the jQuery and other JavaScript libraries hosted by a content distribution network or some other third-party host [90].

The Chrome Web browser handles script embedding in a way that lets us gauge accessibility of *non-script* resources from a remote origin. Chrome fires an `onload` event if it can fetch the resource (*i.e.*, with an `HTTP 200 OK` response), regardless of whether the resource is valid JavaScript. In particular, Chrome respects the `X-Content-Type-Options: nosniff` header, which servers use to instruct browsers to prohibit execution of scripts with an invalid MIME type [10]. Other browsers aren't so forgiving, so we only use this task type on Chrome. Although this technique is convenient, it raises security concerns because other browsers may attempt to execute the fetched object as JavaScript. Section 3.4 describes the infrastructure we use to make this decision.

3.4 Encore Measurement System

This section presents Encore, a distributed platform for measuring Web filtering using the techniques we developed in Section 3.3. Encore selects targets to test for Web filtering (§ 3.4.1), generates measurement tasks to measure those targets (§ 3.4.2), schedules tasks to run on Web clients (§ 3.4.3), delivers these tasks to clients for execution (§ 3.4.4), collects the results of each task (§ 3.4.5), and draws conclusions concerning filtering practices based on the collective outcomes of these tests using the inference techniques that we previously described in Section 3.3.

Figure 6 shows an example of how Encore induces a client to collect measurements of Web filtering. The client visits a Web site `http://example.com`, whose webmaster has volunteered to host Encore. This origin page references a measurement task hosted on a coordination server; the client downloads the measurement task, which in turn instructs the client to attempt to load a resource (*e.g.*, an image) from a measurement target `censored.com`. This request is filtered, so the client informs a collection server of this

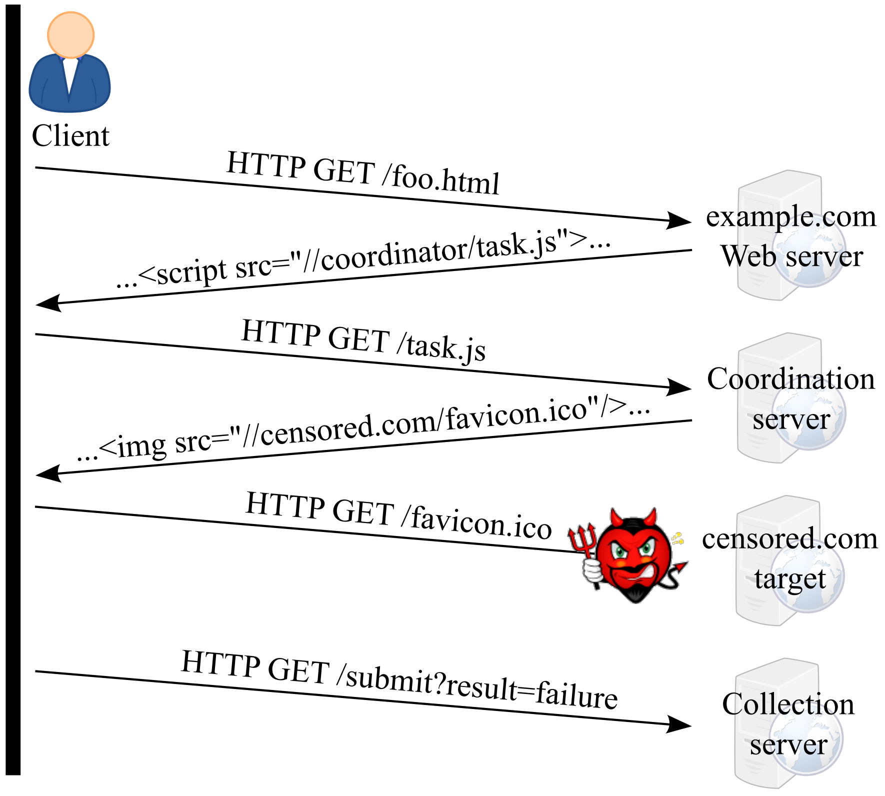


Figure 6: An example of observing Web filtering with Encore. The origin Web page includes Encore's measurement script, which the coordinator decides should test filtering of `censored.com` by attempting to fetch an image. The request for this image fails so the client notifies the collection server of the failure.

filtering. The remainder of this section explains how the origin Web server, coordination server, and collection server work together to induce and collect Web filtering measurements.

3.4.1 Sources of measurement targets

Encore requires a set of potentially filtered Web sites and resources that it will test for Web filtering. This list can contain either specific Web resources (*e.g.*, Web page URLs) in the case that Encore is testing the reachability of a specific page; or a *URL pattern* denoting sets of URLs (*e.g.*, an entire domain name or URL prefix), where Encore is testing the reachability of a domain or a portion of a Web site. A small list of targets with a high chance of filtering is most useful during initial stages of deployment when clients of only a few moderately popular Web sites will likely be contributing measurements. As deployment spreads, a broader set of targets can increase breadth of measurements. We explore both how to obtain lists in both scenarios.

During initial deployment, Encore relies on third parties to provide lists of URLs to test for Web filtering. Several organizations maintain such lists. Some sites rely on per-country experts to curate URLs (*e.g.*, GreatFire for China [77], Filbaan for Iran [65]), while others crowdsource list creation and let anyone contribute reports of Web censorship (*e.g.*, Herdict [82]). Our evaluation in Section 3.5 uses a list of several hundred "high value" URLs curated by Herdict and its partners.

If we deploy Encore to a large number of geographically distributed Web clients, and build a large, accurate Web index, we could instead use Encore clients to verify accessibility of the entire Web index; this would avoid the need for specialized lists of measurement targets by instead testing the entire Web. Regardless of whether Encore curates a small list of high-value measurement targets or simply extracts URLs from a large Web index, these URLs and URL patterns serve as input for Encore's next stage.

Because Encore induces oblivious Web users to perform measurements, there are limitations on the list of targets we will feel comfortable measuring without user consent. Section 3.7 discusses these concerns further.

3.4.2 Generating measurement tasks

Measurement task generation is a three-step procedure that transforms URL patterns from the list of measurement targets into a set of measurement tasks that can determine whether the resources denoted by those URL patterns are filtered for a client. Figure 7 summarizes the process. First, the *Pattern Expander* transforms each URL pattern into a set of URLs by

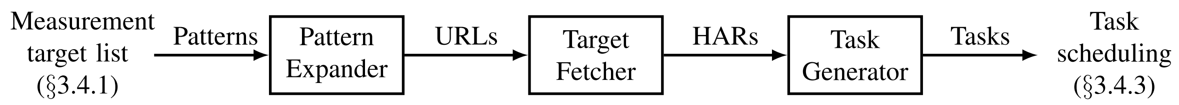


Figure 7: Encore transforms transforms a list of URL patterns to a set of measurement tasks in three steps. A URL pattern denotes a set of URL (*e.g.*, all URLs on a domain). A HAR is an HTTP Archive [80].

searching for URLs on the Web that match the pattern. Second, the *Target Fetcher* collects detailed information about each URL by loading and rendering it in a real Web browser and recording its behavior in an HTTP Archive (HAR) file [80]. Finally, the *Task Generator* examines each HAR file to determine which of Encore's measurement task types, if any, can measure each resource and generates measurement tasks for that subset of resources.

The Pattern Expander searches for URLs that match each URL pattern. This step identifies a set of URLs that can all indicate reachability of a single resource; for example, all URLs with the prefix `http://foo.com/` are candidates for detecting filtering of the `foo.com` domain. Some patterns are trivial (*i.e.*, they match a single URL) and require no work. The remainder require us to discover URLs that match the pattern. We currently expand URL patterns to a sample of up to 50 URLs by scraping site-specific results (*i.e.*, using the *site:* search operator) from a popular search engine. In the future, Encore could use its own Web crawler to explore each pattern.

After expanding URL patterns into a larger set of URLs, the Target Fetcher renders each URL in a Web browser and records a HAR file, which documents the set of resources that a browser downloads while rendering a URL, timing information for each operation, and the HTTP headers of each request and response, among other metadata. We use the PhantomJS [121] headless browser hosted on servers at our university in the United States. To the best of our knowledge, our university does not filter Web requests, especially to the set of URLs we consider in this paper.

Finally, the Task Generator analyzes each HAR file to determine which subset of resources is suitable for measuring using one of the types of measurement tasks from Table 4. It examines timing and network usage of each resource to decide whether a resource is small enough to load from an origin server without significantly affecting user experience, then inspects content type and caching headers to decide whether a resource matches one of the measurement tasks. The Task Generator is particularly conservative when considering inline frames because loading full Web pages can severely impact performance and user experience (*e.g.*, by playing music or videos). Our prototype implementation excludes pages that load flash applets, videos, or any other large objects totalling more than 100 KB, and requires manual verification of pages before deployment; a future implementation could apply stricter controls. Refer back to Section 3.3 for more information on the requirements for each type of measurement task. Section 3.5.1 further explores network overhead of measurement tasks.

3.4.3 Scheduling measurement tasks

After generating measurement tasks, the coordination server must decide which task to schedule on each client. Task scheduling serves two purposes. First, it enables clients to run measurements that meet their restrictions. For example, we should only schedule the script task type from Table 4 on clients running Chrome. In other cases, we may wish to schedule additional tasks on clients that remain idle on an origin Web page for a long time. Second, intelligent task scheduling enables Encore to move beyond analyzing individual measurements and draw conclusions by comparing measurements between clients, countries, and ISPs. For example, a single client in Pakistan could report failure to access a URL for a variety of reasons other than Web filtering (*e.g.*, high client system load, transient DNS failure, WiFi unreliability, etc.). However, if 100 clients measure the same URL within 60 seconds of each other and the only clients that report failure are 10 clients in Pakistan, then we can draw much stronger conclusions about the presence of Web filtering.

3.4.4 Delivering measurement tasks

After scheduling measurement tasks for execution, Encore must deliver tasks to these clients, who subsequently run them and issue cross-origin requests for potentially filtered Web resources. To collect a significant number of useful Web filtering measurements, Encore requires a large client population that is likely to experience a diversity of Web filtering. Previous censorship measurement efforts require researchers to recruit vantage points individually and instruct them to install custom software, which presents a significant deployment barrier [64, 111]. In contrast, Encore recruits a relatively small number of webmasters and piggybacks on top of their sites' existing Web traffic, instantly enlisting nearly *all* of these sites' visitors as measurement collection agents.

A webmaster can enable Encore in several ways. The simplest method is to add a single `<script>` tag that directs clients to load an external JavaScript directly from the coordination server. The coordination server generates a measurement task specific to the client on-the-fly. This method is attractive because it requires no server-side modifications, aside from a single tag; incurs little server overhead (*i.e.*, only the extra time and space required to transmit that single line); and allows the coordination server to tailor measurement tasks to individual clients in real time. Unfortunately, this method is also easiest for censors to fingerprint and disrupt: a censor can simply block access to the coordination server, which inflicts no collateral damage. Section 3.7 discusses ways to make task delivery more robust to blocking, while Section 3.5.3 discusses motivations for webmasters to include Encore on their sites in the first place.

An alternative means of rapidly gaining access to vantage points is to embed measurement tasks inside online advertisements that we purchase. This method exploits webmasters' preestablished relationships with online advertising networks to deliver our measurement tasks without direct consent for measuring Web filtering. Additionally, online ad networks let advertisers sell ads to specific audiences, which will allow us to collect measurements from areas of interest. The main disadvantage of relying entirely on this method for delivering measurement tasks is that there is a single point of failure; the ad network itself may not take kindly to perceived misuse of its service, especially if it leads to network filtering and subsequent loss of revenue in countries wishing to suppress our measurements. During initial deployment of our system, however, these effects are likely to be small enough not to cause trouble.

3.4.5 Collecting measurement results

After clients run a measurement task, they submit the result of the task for analysis. Clients submit the result of task (*i.e.*, whether the client could successfully load the cross-origin resource), related timing information (*i.e.*, how long it took to load the resource), and the task's measurement ID. The process of submitting results is similar to the process that clients use to obtain measurement tasks. In the absence of interference from the adversary, clients submit results by issuing an AJAX request containing the results directly to our collection server. Section 3.7 discusses other ways to submit results if the adversary filters access to the collection server.

3.5 Feasibility of Encore Deployment

We evaluate the feasibility of deploying Encore based on early experience with a prototype implementation and analysis of potential measurement targets. This section addresses three questions about Encore's deployment: (1) whether Encore's measurement tasks can detect filtering of real Web resources, which we explore with offline analysis of potentially-filtered Web sites; (2) whether users visit origin sites, run measurement tasks, and collect measurements, which we estimate using analytics data collected from a likely site of Encore deployment; (3) if webmasters will install Encore, which we study in terms of webmaster incentives and estimated deployment costs.

3.5.1 Can Encore tasks detect filtering?

This section investigates Encore's ability to measure filtering of real Web sites. We investigate feasibility of using Encore to measure filtering of both entire domain names and individual URLs. To measure filtering practices, we use a list of domains and URLs that are "high value" for censorship measurement according to Herdict and its partners [83]; most sites are either perceived as likely filtering targets in many countries (*e.g.*, because they are affiliated with human rights and press freedom organizations) or would cause substantial disruption if filtered (*e.g.*, social media like Twitter and YouTube). This list contains over 200 URL patterns, of which only 178 were online when we performed our analysis in February 2014.

We collect data for this set of experiments by running the first two stages of the pipeline in Figure 7, which uses the Pattern Expander to generate a list of 6,548 URLs from the 178 URL patterns in our list, then collect HAR files for each URL using the Target Fetcher. We then send these HAR files to a modified version of the Task Generator that emits statistics about sizes of accepted resources and pages.

Filtering of entire domains. We explore whether Encore can measure filtering of each of the 178 domains on the list we generated as described above. Recall from Section 3.3.3 that we can use either images or style sheets to observe Web filtering of an entire domain; for simplicity, this analysis only considers images, although style sheets work similarly. We can measure a domain using this technique if it (1) contains images that can be embedded by an origin site and (2) those images are small enough not to significantly impact user experience. We explore both of these requirements for the 178 domains in our list. Because our prototype implementation expands URL patterns using the top 50 search results for that pattern, we will be analyzing a sample of at most 50 URLs per domain. Most of these domains have more than 50 pages, so our results are a lower bound of the amenability of Encore to collect censorship measurements from each domain.

Figure 8 plots the distribution of the number of images that each domain hosts. 70% of domains embed at least one image, and almost all such images are less than 5 KB. Nearly as many domains embed images that fit within a single packet and a third of domains have hundreds of such images. Even if we conservatively restrict measurement tasks to load images less than 1 KB, Encore can effectively measure Web filtering of over half the domains in our list.

Filtering of specific Web sites. We explore how often Encore can measure filtering of

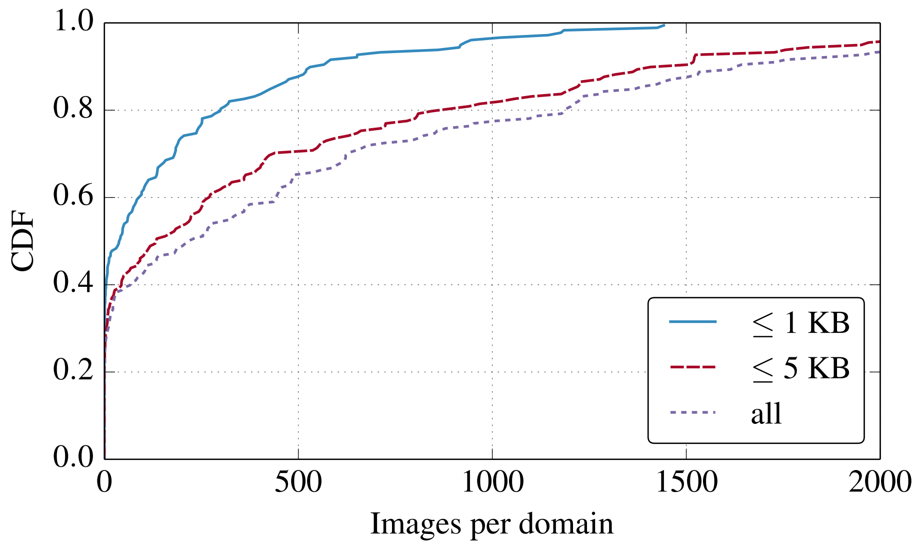


Figure 8: Distribution of the number of images hosted by each of the 178 domains tested, for images that are at most 1 KB, at most 5 KB, and any size. Over 60% of domains host images that could be delivered to clients inside a single packet, and a third of domains have hundreds of such images to choose from.

individual URLs hosted on the domains from the list we constructed. Table 4 presented three mechanisms for measuring filtering of individual Web pages: two iframe mechanisms and a browser-specific script embedding mechanism. We explore the feasibility the "inline frames (caching)" mechanism, which attempts to load a Web page in an iframe, then verifies that the browser cached embedded resources from that page. We can use this mechanism to measure filtering of pages that (1) don't incur too much network overhead when loading in a hidden iframe and (2) embed cacheable images. Analysis of the "inline frames (load time)" mechanism requires complex parameter tuning that requires a lot of timing from a diverse client population. Although the "script embedding" mechanism works with nearly all URLs, the technique only works for users who are browsing using Chrome.

We first study the expected network overhead from loading sites in an iframe. Figure 9 plots the distribution of page sizes for each URL, where the page size is the sum of sizes of all resources a page loads and is a rough lower bound on the network overhead that would be incurred by loading each page in a hidden iframe (protocol negotiation and inefficiencies add further overhead). Page sizes are distributed relatively evenly between 0–2 MB with a very long tail. Our prototype only permits measurement tasks to load pages smaller than 100 KB, although future implementations could perhaps tune this bound depending on a client's performance and preferences.

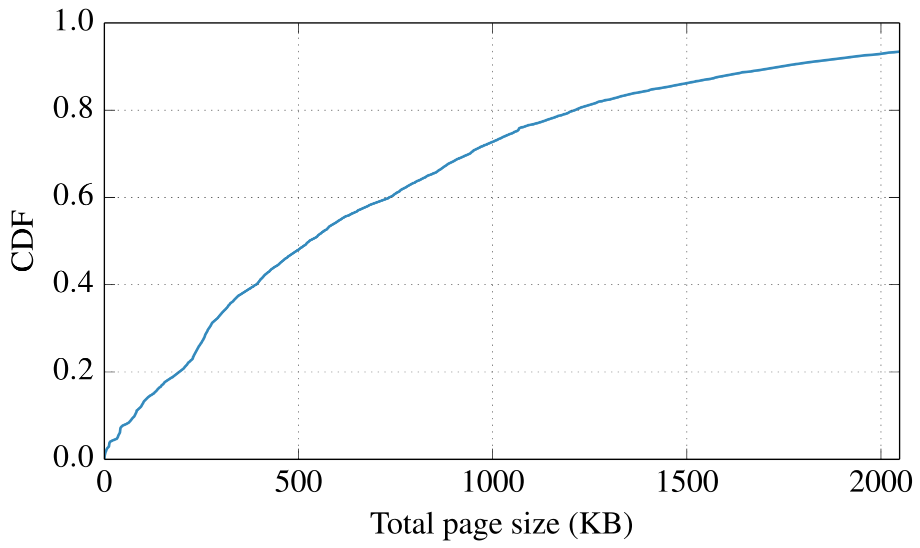


Figure 9: Distribution of page sizes, computed as the sum of sizes of all objects loaded by a page. This indicates the network overhead each page would incur if a measurement task loaded it in a hidden iframe. Over half of pages load at least half a megabyte of objects.

We then evaluate whether these sites embed content that can be retrieved with cross-origin requests. Figure 10 shows the distribution of the number of cacheable images per URL for pages that are at most 100 KB, at most 500 KB, and any size. Nearly 70% of pages embed at least one cacheable image and half of pages cache five or more images, but these numbers drop significantly when restricting page sizes. Only 30% of pages at most 100 KB embed at least one cacheable image.

Encore can measure filtering of upwards of 50% of domains depending on the sizes of images, but less than 10% of URLs when we limit pages to 100 KB. This finding supports our earlier observation in Section 3.3.3 that detecting the filtering of individual Web resources may be significantly more difficult than detecting the filtering of entire domains.

3.5.2 Who performs Encore measurements?

Encore requires clients to visit the Web sites that are hosting Encore scripts. The demographics of clients who perform Encore measurements is closely related to the demographics of the clients who visit a participating Web site. To evaluate whether a typical Web site would receive measurements from enough locations, we installed Encore on the home page of a university faculty member for February 2014 and analyzed the measurement results and analytics data it collected. We ran a single measurement task that loads an icon from `google.com`. Over the course of the month, the site saw $1,171$ visits. Most visitors were

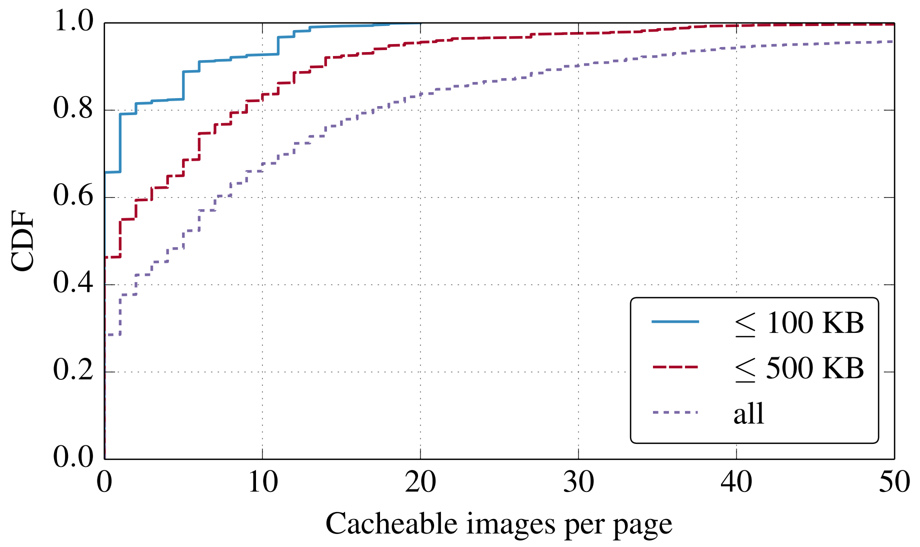


Figure 10: Distribution of the number of cacheable images loaded by pages that require at most 100 KB of traffic to load, pages that incur at most 500 KB of traffic, and all pages. Perhaps unsurprisingly, smaller pages contain fewer (cacheable) images. Over 70% of all pages cache at least one image and half of all pages cache five or more images; these numbers drop considerably when excluding pages greater than 100 KB.

from the United States, but we saw more than 10 users from 10 other countries, and 16% of visitors reside in countries with well-known Web filtering policies (India, China, Pakistan, the UK, and South Korea), indicating that dispatching measurement tasks to sites such as academic Web pages may actually result in measurements from a variety of representative locations.

Of the $1,171$ unique visitors to the site, 999 attempted to run a measurement task (we confirmed nearly all of the rest to be automated traffic from our campus' security scanner), and 997 successfully fetched a cross-origin resource. Only one reported a failure, and one didn't report a result either way. The single failure came from a client in India, which might have poor connectivity even to popular services like Google. We also found that 45% of visitors remained on the page for longer than 10 seconds, which is more than sufficient time to execute at least one measurement task and report its results. The 35% of visitors who remained for longer than a minute could actually run multiple measurement tasks.

Our small pilot deployment of Encore is representative of the sites on which we can expect Encore to be deployed in the short term. Our next round of webmaster recruitment will likely be to friends and colleagues who are mostly academics and whose Web sites and whose audience likely resembles our pilot site. Although adoption of Encore by even a single high-traffic Web site would entirely eclipse measurements collected by these small university deployments, grassroots recruitment remains necessary: Encore relies on a variety of origin sites to deter an adversary from simply blocking access to all origins to suppress our measurement collection. Section 3.7 discusses further mechanisms for deterring filtering of Encore's origin sites and backend infrastructure.

3.5.3 Will webmasters install Encore?

Encore cannot directly target specific demographics for measurement collection—the measurements that we collect arise "organically" from the set of users who happen to visit a Web site that has installed an Encore script. If the sites that host Encore are globally popular sites (*e.g.*, Google), then Encore can achieve an extremely widespread sampling of users. If, on the other hand, the sites are only popular in particular regions, the resulting measurements will be more limited. The best solution to this problem is to achieve deployment across sites that are likely to be popular among client populations where we would like measurements or to deploy on globally popular sites.

Recruiting webmasters to include Encore's measurement scripts should be feasible. First, installing Encore on a Web site incurs little cost. Serving these scripts to clients adds minimal network overhead; our prototype adds only 100 bytes to each origin page and requires no additional requests or connections between the client and the origin server. Measurements themselves have little effect on the Web page's perceived performance because they run asynchronously after the page has loaded and rendered. However, they do incur some network overhead to clients when loading cross-origin resources, which may be undesirable to users with bandwidth caps or slow, shared network connections. As Section 3.5.1 explained, measurement tasks that detect filtering of a domain (*i.e.*, by loading small images) incur overheads that are an insignificant fraction of a page's network usage.

Second, we see two strong incentives for webmasters to participate in Encore. Many webmasters may support Encore simply out of greater interest in measuring Web filtering and encouraging transparency of government censorship. The grassroots success of similar online freedom projects (*e.g.*, Tor) in recruiting people to host relays and bridges suggests that such a population does exist. For further incentive, we could institute a reciprocity agreement for webmasters: in exchange for installing our measurement scripts, webmasters could add their own site to Encore's list of targets and receive notification about their site's availability from Encore's client population.

3.6 Measurements

In this section, we confirm the soundness of Encore's measurement tasks with both controlled experiments and by comparing our ability to confirm cases of Web filtering with independent reports of filtering from other research studies. We have implemented and released every component of Encore described in Section 3.4 and have collected one month of measurements in April 2014 [55].

To date, eight volunteers have deployed Encore on their sites; most of our measurements originate from the home pages of two professors who receive high traffic volumes. We recorded 6,666 measurements from 3,347 distinct IPs in 98 countries, with the United States, China, Canada, India, Germany, Brazil, the United Kingdom, Egypt, and South Korea each reporting more than 100 measurements. Most of these countries practice some form of Web filtering. We use a standard IP geolocation database to determine client locations [99].

3.6.1 Are measurement tasks sound?

To confirm the soundness of Encore's measurements, we built a Web censorship testbed, which has DNS, firewall, and Web server configurations that emulate seven varieties of DNS, IP, and HTTP filtering [156]. During the latter half of the month, we instructed approximately 30% of clients to measure resources hosted by the testbed (or unfiltered control resources) using the four task types from Table 4. For example, we verified that the *images* task type detects DNS blocking by attempting to load an image from an invalid domain and observing that the task reports filtering; we verified that the same task successfully loads an unfiltered image.[1]

This verification is straightforward for the image, style sheet, and script task types because they give explicit binary feedback about whether a resource successfully loaded. Encore collected 930 testbed measurements for these task types; after excluding erroneously contributed measurements (*e.g.*, from Web crawlers), there were no true positives and few false positives. For example, clients in India, a country with notoriously unreliable network connectivity, contributed to a 5% false positive rate for images.

Verifying soundness of the inline frame task type requires more care because it infers existence of filtering from the time taken to load resources. Figure 11 compares the time taken to load an uncached versus cached single pixel image from 42 random globally distributed Encore clients. Cached images normally load within a few tens of milliseconds,

[1]From `//invalid.noise.gatech.edu/pixel.png` and `//encore.noise.gatech.edu:8892/pixel.png`, respectively.

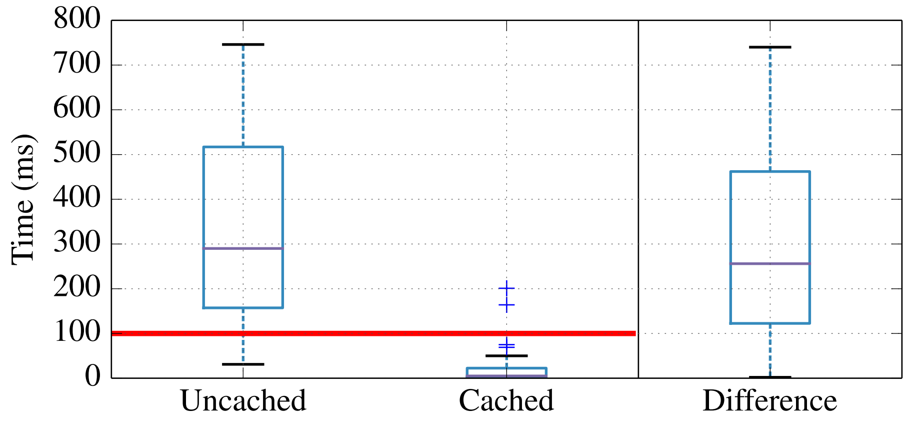


Figure 11: Comparison between load times for cached and uncached images from 42 Encore clients. Cached images typically load within tens of milliseconds, whereas uncached usually take at least 100 ms longer to load, indicated by the bold red line. We use this difference to infer filtering.

whereas most clients take at least 100 ms longer to load the same image uncached. The few clients with little difference between cached and uncached load time were located at our university, very close to the server. Difference in load time will be more pronounced for larger images and with greater latency between clients and content.

In both cases, false positives highlight (1) that distinguishing Web filtering from other kinds of network problems is difficult and (2) the importance of collecting many measurements before drawing strong conclusions about Web filtering. We now develop a filtering detection algorithm that addresses both concerns.

3.6.2 Does Encore detect Web filtering?

We instructed the remaining 70% of clients to measure resources suspected of filtering, with the goal of independently confirming Web filtering in countries that other research has reported. We aimed to detect resources that are consistently inaccessible from one region, yet still accessible from others. For this purpose, we only measure filtering of entire domains using the image task type.

Detecting Web filtering is challenging because measurement tasks may fail for reasons other than Web filtering: clients may experience intermittent network connectivity problems, browsers may incorrectly execute measurement tasks, sites may themselves go offline, and so on. We use a statistical hypothesis test to distinguish such sporadic or localized measurement failures from more consistent failures that might indicate Web filtering. We model each measurement success as a Bernoulli random variable with parameter

$p = 0.7$; we assume that, in the absence of Web filtering, clients should successfully load resources at least 70% of the time. Although this quite a conservative assumption, it captures our desire to eliminate false positives, which can easily drown out true positives when detecting rare events like Web filtering. For each resource and region, we count both the total number of measurements n_r and the number of successful measurements x_r and run a one-sided hypothesis test for a binomial distribution; we consider a resource as filtered in region r if x_r fails this test at 0.05 significance (*i.e.*, $\Pr[\mathsf{Binomial}(n_r, p) \leq x_r] \leq 0.05$) yet does not fail the same test in other regions.

Applying this technique on preliminary measurements confirms well-known censorship of `youtube.com` in Pakistan, Iran, and China [76], and filtering of several sites in China, including `twitter.com`, `anonymouse.org` (a filtering circumvention tool) and `pastebin.com` (a site that lets users share snippets of text) [77]. Measurements also appear to confirm Turkey's widely-publicized filtering of YouTube at the end of March [145], but we don't yet have enough measurements for statistical significance. We are actively recruiting more sites and collecting more measurement, and continue to expand these results. Although our detection algorithm works well on preliminary data, possible enhancements include dynamically tuning model parameters to account for differing false positive rates in each country and accounting for potential confounding factors like behavior differences between browsers, operating systems, and ISPs [15].

3.7 Security Considerations

We discuss how Encore operates in the presence of an adversary that can block, disrupt, or tamper with client measurements or collection infrastructure. We discuss several tactics that can mitigate attacks, but Encore ultimately faces fundamental security limitations because it operates within Web browsers, which have limited security APIs and may themselves have security vulnerabilities. In particular, because Encore cannot establish trust of clients performing measurements, it can do little to prevent an adversary from contributing false measurements; any system accepting measurements from anonymous or pseudonymous clients faces this risk.

Filtering access to Encore infrastructure. Clients can only use Encore if they can fetch a measurement task. If the domain (or URL) that hosts measurement tasks is itself blocked, clients will not be able to execute measurements. Once a client retrieves a measurement task, subsequent requests appear as ordinary cross-origin requests; as a result, the main concern is ensuring that clients can retrieve measurement tasks in the first place.

The server that dispatches tasks could be replicated across many domains to make it more difficult for a censor to block Encore by censoring a single domain. Clients could contact the coordination server indirectly via an intermediary or create mirrors of the coordination server in shared hosting environments (*e.g.*, Amazon AWS), thereby increasing the collateral damage of blocking a mirror. Going further, webmasters could contact the coordination server on behalf of clients (*e.g.*, with a WordPress plugin or Django package) by querying the coordination server and including the returned measurement task directly in the page it serves to the client; to increase scalability and decrease latency, servers could cache several tasks in advance. Similarly, collection of the results could be distributed across servers hosted in different domains, to ensure that collection is not blocked.

There are limits to Encore's ability to withstand such attacks. Because it runs entirely within a Web browser, Encore cannot leverage stronger security tools like Tor to anonymously report measurements as done by other censorship measurement tools [51, 158].

Detecting and interfering with Encore measurements. Blocking Encore based on the contents of measurement tasks (*e.g.*, via deep packet inspection) should be difficult, because we can easily disguise tasks' code using JavaScript obfuscation or detection evasion techniques [53, 87]. Identifying task behavior is equally difficult because it appears merely as requests to load a cross-origin object—something many Web sites do under normal operation. If a single client performs a *sequence* of cross-origin requests that appear unrelated to the content of the host site, a censor may recognize the sequence as unusual and either block the subsequent reports or otherwise attempt to distort the results. We expect such interference to be relatively difficult, however, since a censor would first have to identify a sequence of requests as a measurement attempt and interpose on subsequent requests to interfere with the reports. Although such interference is plausible, censors do not generally interfere with measurements today, so we leave this consideration to future work.

User privacy and protection. Encore induces clients to perform Web requests to potentially sensitive or censored Web sites, which could prove incriminating to the client in certain countries and circumstances. Our conversations with citizens of various countries suggest that attempting to access censored content is not illegal *per se*; nevertheless, simply because an activity is legal does not make it harmless, so we are sensitive to inducing users to retrieve content that might land them in trouble. Because Encore's measurements are effectively a side effect of a user visiting an unrelated host site, every client has plausible deniability; most users are unaware of Encore, which runs in the background and has no visible effects. Encore could also shield clients from more incriminating content—for example, loading a small icon from a banned site may be less incriminating than loading actual content.

Although our work doesn't prescribe a specific use case, we recognize that developing and deploying a tool like Encore entails risks that we need to understand better; we are working with experts in Internet policy to find useful yet safe deployment scenarios for Encore.

3.8 Conclusion

Despite the importance of measuring the extent and nature of Internet censorship, doing so is difficult because it requires deploying a large number of geographically diverse vantage points, and recruiting volunteers for such a deployment is a significant deployment hurdle. This chapter presents an alternate approach: rather than requiring users to install custom measurement software, we take advantage of the fact that users' Web browsers can perform certain types of cross-origin requests, which can harness to induce measurements of reachability to arbitrary third-party domains. Although only a limited amount of information about the success of these requests leaks across domains, even a small amount of leakage turns out to be enough to permit inferences about the reachability of higher-level Web resources, including both URL prefixes and even entire domains.

Encore shifts the deployment burden from clients to webmasters. We have designed Encore so that deployment is simple (in many cases, webmasters only add one line to the main Web page source). We also point out that many webmasters are typically interested in monitoring the reachability of their sites from various client geographies and networks in any case, so deployment incentives are well-aligned.

Although the types of measurements Encore can perform may be more definitive than tools that rely on informal user reports (*e.g.*, Herdict), Encore may draw far fewer conclusions about the scope and methods of censorship than client-hosted measurement tools that are specifically designed to measure censorship methods in detail (*e.g.*, OONI, Censorscope). Ultimately, censorship measurement is a complex, moving target, and no single measurement method or tool can paint a complete picture of censorship practices. What is sorely missing from the existing set of measurement tools, however, is a way to characterize censorship practices in broad strokes, based on a sizeable and continuous set of client measurements. By filling this important hole in our understanding, Encore can help bridge the gap between diverse yet inconclusive user reports and detailed yet narrow or short-term fine-grained measurements.

CHAPTER IV

CIRCUMVENTING CENSORSHIP WITH USER-GENERATED CONTENT

4.1 Introduction

Although existing anti-censorship systems—notably, onion routing systems such as Tor [51]—have allowed citizens some access to censored information, these systems require users outside the censored regime to set up infrastructure: typically, they must establish and maintain proxies of some kind. The requirement for running fixed infrastructure outside the firewall imposes two limitations: (1) a censor can discover and block the infrastructure; (2) benevolent users outside the firewall must install and maintain it, a potentially arduous task. As a result, these systems are somewhat easy for censors to monitor and block and can be hard to maintain at scale. Although proxy systems like Tor continue to enjoy widespread use, governments have become increasingly adept at blocking it, begging the question of whether there are fundamentally new approaches to advancing the arms race between censors and users: specifically, we explore whether it is possible to circumvent censorship firewalls with infrastructure that is more pervasive, and that does not require individual users or organizations to maintain it.

We realize that countless sites allow users to upload content to infrastructure that they do not maintain or own through a variety of media, ranging from photos to blog comments to videos. Leveraging the large number of sites that allow users to upload their own content potentially yields many small cracks in censorship firewalls, because there are many different types of media that users can upload, and many different sites where they can upload it. The sheer number of sites that users could use to exchange messages, and the many different ways they could hide content, makes it difficult for a censor to successfully monitor and block all of them.

In this chapter, we design a system to circumvent censorship firewalls using different types of user-generated content as cover traffic. We present Collage, a method for building message channels through censorship firewalls using user-generated content as the cover medium. Collage uses existing sites to host user-generated content that serves as the cover for hidden messages (*e.g.*, photo-sharing, microblogging, and video-sharing sites). Hiding messages in photos, text, and video across a wide range of host sites makes it more difficult for censors to block all possible sources of censored content. Second, because the messages are hidden in other seemingly innocuous messages, Collage provides its users some level of deniability that they do not have in using existing systems (*e.g.*, accessing a Tor relay node immediately implicates the user that contacted the relay). We can achieve these goals with minimal out-of-band communication.

Collage is not the first system to suggest using covert channels: much previous work has explored how to build a covert channel that uses images, text, or some other media as cover traffic, sometimes in combination with mix networks or proxies [5,20,46,51,59,116,136]. Other work has also explored how these schemes might be broken [71], and others hold the view that message hiding or "steganography" can never be fully secure. Collage's new contribution, then, is to design covert channels based on user-generated content and imperfect message-hiding techniques in a way that circumvents censorship firewalls that is robust enough to allow users to freely exchange messages, even in the face of an adversary that may be looking for such suspicious cover traffic.

The first challenge in designing Collage is to develop an appropriate *message vector* for embedding messages in user-generated content. Our goal for developing a message vector is to find user-generated traffic (*e.g.*, photos, blog comments) that can act as a cover medium, is widespread enough to make it difficult for censors to completely block and remove, yet is common enough to provide users some level of deniability when they download the cover traffic. In this chapter, we build message vectors using the user-generated photo-sharing site, Flickr [66], and the microblogging service, Twitter [147], although our system in no way depends on these particular services. We acknowledge that some or all of these two specific sites may ultimately be blocked in certain countries; indeed, we witnessed that parts of Flickr were already blocked in China when accessed via a Chinese proxy in January 2010. A main strength of Collage's design is that blocking a specific site or set of sites will not fully stem the flow of information through the firewall, since users can use so many sites to post user-generated content. We have chosen Flickr and Twitter as a proof of concept, but Collage users can easily use domestic equivalents of these sites to communicate using Collage.

Given that there are necessarily many places where one user might hide a message for another, the second challenge is to agree on *rendezvous* sites where a sender can leave a message for a receiver to retrieve. We aim to build this *message layer* in a way that the client's traffic looks innocuous, while still preventing the client from having to retrieve an unreasonable amount of unwanted content simply to recover the censored content. The basic idea behind rendezvous is to embed message segments in enough cover material so that it is difficult for the censor to block all segments, even if it joins the system as a user; and

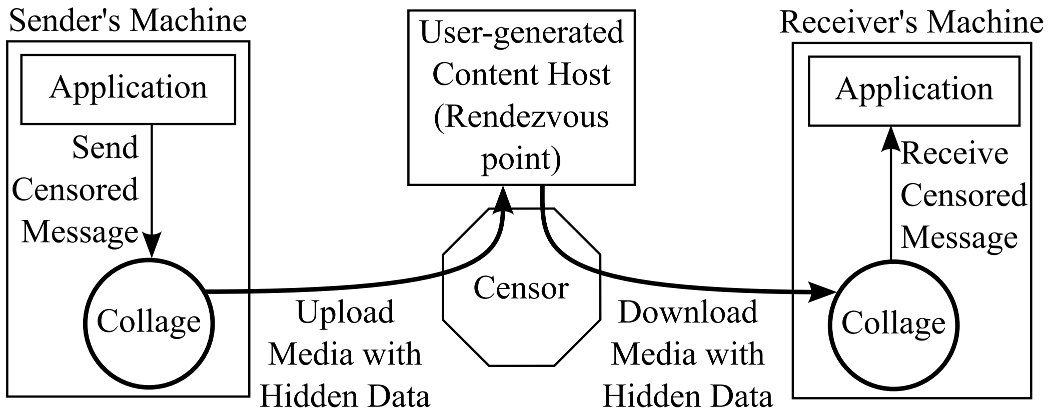


Figure 12: Collage's interaction with the network. See Figure 13 for more detail.

users can retrieve censored messages without introducing significant deviations in their traffic patterns. In Collage, senders and receivers agree on a common set of network locations where any given content should be hidden; these agreements are established and communicated as "tasks" that a user must perform to retrieve the content (*e.g.*, fetching a particular URL, searching for content with a particular keyword). Figure 12 summarizes this process. Users send a message with three steps: (1) divide a message into many erasure-encoded "blocks" that correspond to a task, (2) embed these blocks into user-generated content (*e.g.*, images), and (3) publish this content at user-generated content sites, which serve as rendezvous points between senders and receivers. Receivers then retrieve a subset of these blocks to recover the original message by performing one of these tasks.

This chapter makes the following contributions.

- We present the design and implementation of Collage, a censorship-resistant message channel built using user-generated content as the cover medium. An implementation of the Collage message channel is publicly available [36].
- We evaluate the performance and security of Collage. Collage does impose some overhead, but the overhead is acceptable for small messages (*e.g.*, articles, emails, short messages), and Collage's overhead can also be reduced at the cost of making the system less robust to blocking.
- We present Collage's general message-layer abstraction and show how this layer can serve as the foundation for two different applications: Web publishing and direct messaging (*e.g.*, email). We describe and evaluate these two applications.

The rest of this chapter proceeds as follows. In Section 4.2, we describe the design goals for Collage and the capabilities of the censor. Section 4.2.3 discusses prior message hiding techniques that inspire Collage. Section 4.3 presents the design and implementation of Collage. Section 4.4 evaluates the performance of Collage's messaging layer and applications. Section 4.5 describes the design and implementation of two applications that are built on top of this messaging layer. Section 4.6 discusses some limitations of Collage's design and how Collage might be extended to cope with increasingly sophisticated censors. Section 4.7 concludes.

4.2 Problem Overview

We now discuss our model for the censor's capabilities and our goals for circumventing a censor who has these capabilities. Whereas Chapter 2 introduced details on how censors have implemented censorship in the past, the goal of this section is to try to account for future censorship mechanisms. That said, it is difficult, if not impossible, to fully determine the censor's potential, or event current, capabilities; as a result, Collage cannot provide formal guarantees regarding success or deniability. Instead, we present a model for the censor that we believe is more advanced than current capabilities and, hence, where Collage is *likely* to succeed. Nevertheless, censorship is an arms race, so as the censor's capabilities evolve, attacks against censorship firewalls will also need to evolve in response. In Section 4.6, we discuss how Collage's could be extended to deal with these more advanced capabilities as the censor becomes more sophisticated.

We note that although we focus on censors, Collage also depends on content hosts to store media containing censored content. Content hosts currently do not appear to be averse to this usage (*e.g.*, to the best of our knowledge, Collage does not violate the Terms of Service for either Flickr or Twitter), although if Collage were to become very popular this attitude would likely change. Although we would prefer content hosts to willingly serve Collage content (*e.g.*, to help users in censored regimes), Collage can use many content hosts to prevent any single host from compromising the entire system.

4.2.1 The Censor

We assume that the censor wishes to allow *some* Internet access to clients, but can monitor, analyze, block, and alter subsets of this traffic. We believe this assumption is reasonable: if the censor builds an entirely separate network that is partitioned from the Internet, there is little we can do. Beyond this basic assumption, there is a wide range of capabilities we can assume. Perhaps the most difficult aspect of modeling the censor is figuring out how much effort it will devote to capturing, storing, and analyzing network traffic. Our model assumes that the censor can deploy monitors at multiple network egress points and observe all traffic as it passes (including both content and headers). We consider two types of capabilities:

targeting and disruption.

Targeting. A censor might *target* a particular user behind the firewall by focusing on that user's traffic patterns; it might also target a particular suspected content host site by monitoring changes in access patterns to that site (or content on that site). In most networks, a censor can monitor all traffic that passes between its clients and the Internet. Specifically, we assume the censor can eavesdrop any network traffic between clients on its network and the Internet. A censor's motive in passively monitoring traffic would most likely be either to determine that a client was using Collage or to identify sites that are hosting content. To do so, the censor could monitor traffic *aggregates* (*i.e.*, traffic flow statistics, like NetFlow [106]) to determine changes in overall traffic patterns (*e.g.*, to determine if some website or content has suddenly become more popular). The censor can also observe traffic streams from *individual* users to determine if a particular user's clickstream is suspicious, or otherwise deviates from what a real user would do. These capabilities lead to two important requirements for preserving deniability: traffic patterns generated by Collage should not skew overall distributions of traffic, and the traffic patterns generated by an individual Collage user must resemble the traffic generated by innocuous individuals.

To target users or sites, a censor might also use Collage as a sender or receiver. This assumption makes some design goals more challenging: a censor could, for example, inject bogus content into the system in an attempt to compromise message availability. It could also join Collage as a client to discover the locations of censored content, so that it could either block content outright (thus attacking availability) or monitor users who download similar sets of content (thus attacking deniability). We also assume that the censor could act as a content publisher. Finally, we assume that a censor might be able to coerce a content host to shut down its site (an aggressive variant of actively blocking requests to a site).

Disruption. A censor might attempt to *disrupt* communications by actively mangling traffic. We assume the censor would not mangle uncensored content in any way that a user would notice. A censor could, however, inject additional traffic in an attempt to confuse Collage's process for encoding or decoding censored content. We assume that it could also block traffic at granularities ranging from an entire site to content on specific sites.

The costs of censorship. In accordance with Bellovin's observations [14], we assume that the censor's capabilities, although technically limitless, will ultimately be constrained by cost and effort. In particular, we assume that the censor will *not* store traffic indefinitely, and we assume that the censor's will or capability to analyze traffic prevents it from observing more complex statistical distributions on traffic (*e.g.*, we assume that it cannot perform analysis based on joint distributions between arbitrary pairs or groups of users). We also assume that the censor's computational capabilities are limited: for example, performing deep packet inspection on every packet that traverses the network or running statistical analysis against all traffic may be difficult or infeasible, as would performing sophisticated timing attacks (*e.g.*, examining inter-packet or inter-request timing for each client may be computationally infeasible or at least prohibitively inconvenient). As the censorship arms race continues, the censor may develop such capabilities.

4.2.2 Circumventing the Censor

Our goal is to allow users to send and receive messages across a censorship firewall that would otherwise be blocked; we want to enable users to communicate across the firewall by exchanging articles and short messages (*e.g.*, email messages and other short messages). In some cases, the sender may be behind the firewall (*e.g.*, a user who wants to publish an article from within a censored regime). In other cases, the receiver might be behind the firewall (*e.g.*, a user who wants to browse a censored website).

We aim to understand Collage's performance in real applications and demonstrate that it is "good enough" to be used in situations where users have no other means for circumventing the firewall. We therefore accept that our approach may impose substantial overhead, and we do not aim for Collage's performance to be comparable to that of conventional networked communication. Ultimately, we strive for a system that is effective and easy to use for a variety of networked applications. To this end, Collage offers a messaging library that can support these applications; Section 4.5 describes two example applications.

Collage's main performance requirement is that the overhead should be small enough to allow content to be stored on sites that host user-generated content and to allow users to retrieve the hidden content in a reasonable amount of time (to ensure that the system is usable), and with a modest amount of traffic overhead (since some users may be on connections with limited bandwidth). In Section 4.4, we evaluate Collage's storage requirements on content hosting sites, the traffic overhead of each message (as well as the tradeoff between this overhead and robustness and deniability), and the overall transfer time for messages.

In addition to performance requirements, we want Collage to be robust in the face of the censor that we have outlined in Section 4.2.1. We can characterize this robustness in terms of two more general requirements. The first requirement is *availability*, which says that clients should be able to communicate in the face of a censor that is willing to restrict access to various content and services. Most existing censorship circumvention systems do not prevent a censor from blocking access to the system altogether. Indeed, regimes such as China have blocked or hijacked applications ranging from websites [139] to peer-to-peer systems [144] to Tor itself [142]. We aim to satisfy availability in the face of the censor's targeting capabilities that we described in Section 4.2.1.

Second, Collage should offer users of the system some level of *deniability*; although this design goal is hard to quantify or formalize, informally, deniability says that the censor cannot discover the users of the censorship system. It is important for two reasons. First, if the censor can identify the traffic associated with an anti-censorship system, it can discover and either block or hijack that traffic. As mentioned above, a censor observing encrypted traffic may still be able to detect and block systems such as Tor [51]. Second, and perhaps more importantly, if the censor can identify specific users of a system, it can coerce those users in various ways. Past events have suggested that censors are able and willing to both discover and block traffic or sites associated with these systems *and* to directly target and punish users who attempt to defeat censorship. In particular, China requires users to register with ISPs before purchasing Internet access at either home or work, to help facilitate tracking individual users [32]. Freedom House reports that in six of fifteen countries they assessed, a blogger or online journalist was sentenced to prison for attempting to circumvent censorship laws—prosecutions have occurred in Tunisia, Iran, Egypt, Malaysia, and India [68]—and cites a recent event of a Chinese blogger who was recently attacked [33]. As these regimes have indicated their willingness and ability to monitor and coerce individual users, we believe that attempting to achieve some level of deniability is important for any anti-censorship system.

By design, a user cannot disprove claims that he engages in deniable communication, thus making it easier for governments and organizations to implicate arbitrary users. We accept this as a potential downside of deniable communications, but point out that organizations can already implicate users with little evidence (*e.g.*, [2]).

4.2.3 Prior message hiding and embedding techniques

Collage relies on techniques that can embed content into cover traffic. The current implementation of Collage uses an image steganography tool called `outguess` [116] for hiding content in images and a text steganography tool called SNOW [136] for embedding content in text. We recognize that steganography techniques offer no formal security guarantees; in fact, these schemes can and have been subject to various attacks (*e.g.*, [71]). Danezis has also noted the difficulty in building covert channels with steganography alone [43]: not only can the algorithms be broken, but also they do not hide the identities of the communicating parties. Thus, these functions must be used as components in a larger system, not as standalone "solutions". Collage relies on the embedding functions of these respective algorithms, but its security properties do not hinge solely on the security properties of

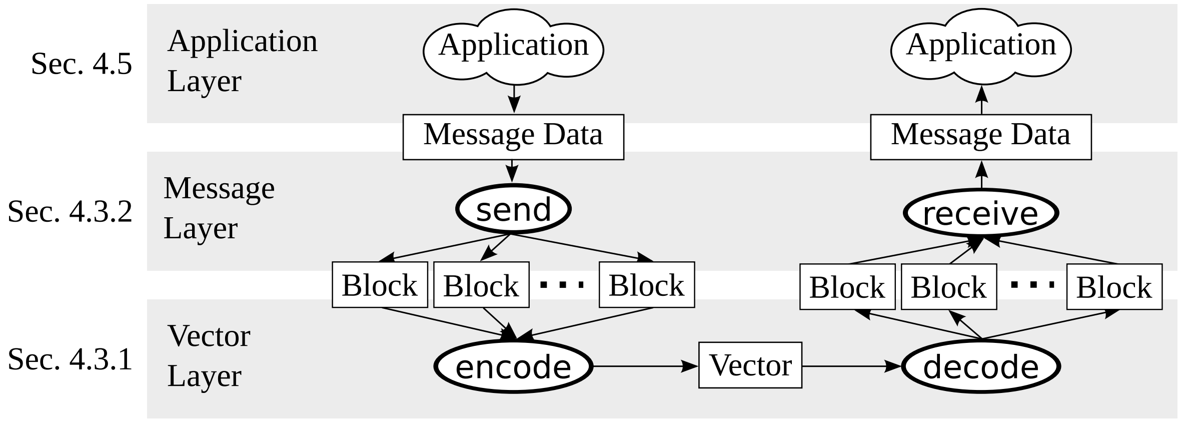


Figure 13: Collage's layered design model. Operations are in bolded rounded rectangles; intermediate data forms are in rectangles.

any single information hiding technique; in fact, Collage could have used watermarking techniques instead, but we chose these particular embedding techniques for our proof of concept because they had readily available, working implementations. One of the challenges that Collage's design addresses is how to use imperfect message hiding techniques to build a message channel that is both available and offers some amount of deniability for users.

4.3 Collage Design and Implementation

Collage's design has three layers and roughly mimics the layered design of the network protocol stack itself. Figure 13 shows these three layers: the vector, message, and application layers. The *vector layer* provides storage for short data chunks (Section 4.3.1), and the *message layer* specifies a protocol for using the vector layer to send and receive messages (Section 4.3.2). A variety of applications can be constructed on top of the message layer. We now describe the vector and message layers in detail, deferring discussion of specific applications to Section 4.5. After describing each of these layers, we discuss *rendezvous*, the process by which senders and receivers find each other to send messages using the message layer (Section 4.3.3). Finally, we discuss our implementation and initial deployment (Section 4.3.4).

4.3.1 Vector Layer

The vector layer provides a substrate for storing short data chunks. Effectively, this layer defines the "cover media" that should be used for embedding a message. For example, if a small message is hidden in the high frequency of a video then the *vector* would be, for example, a YouTube video. This layer hides the details of this choice from higher layers and exposes three operations: encode, decode, and isEncoded. These operations encode data into a vector, decode data from an encoded vector, and check for the presence of encoded data given a secret key, respectively.

Collage imposes requirements on the choice of vector. First, each vector must have some capacity to hold encoded data. Second, the population of vectors must be large so that many vectors can carry many messages. Third, to satisfy both availability and deniability, it must be relatively easy for users to deniably send and receive vectors containing encoded chunks. Fourth, to satisfy availability, it must be expensive for the censor to disrupt chunks encoded in a vector. Any vector layer with these properties will work with Collage's design, although the deniability of a particular application will also depend upon its choice of vector, as we discuss in Section 4.6.

The feasibility of the vector layer rests on a key observation: *data hidden in user-generated content serves as a good vector for many applications, since it is both populous and comes from a wide variety of sources (*i.e.*, many users).* Examples of such content include images published on Flickr [66] (as of June 2009, Flickr had about 3.6 billion images, with about 6 million new images per day [72]), tweets on Twitter [147] (Twitter had about half a million tweets per day [150], and Mashable projected about 18 million Twitter users by the end of 2009 [148]), and videos on YouTube [166], which had about $200,000$ new videos per day as of March 2008 [167].

For concreteness, we examine two classes of vector encoding algorithms. The first option is *steganography*, which attempts to hide data in a cover medium such that only intended recipients of the data (*e.g.*, those possessing a key) can detect its presence. Steganographic techniques can embed data in a variety of cover media, such as images, video, music, and text. Steganography makes it easy for legitimate Collage users to find vectors containing data and difficult for a censor to identify (and block) encoded vectors. Although the deniability that steganography can offer is appealing, key distribution is challenging, and almost all production steganography algorithms have been broken. Therefore, we cannot simply rely on the security properties of steganography.

Another option for embedding messages is *digital watermarking*, which is similar to steganography, except that instead of hiding data from the censor, watermarking makes it difficult to remove the data without destroying the cover material. Data embedded using watermarking is perhaps a better choice for the vector layer: although encoded messages are clearly visible, they are difficult to remove without destroying or blocking a large amount of legitimate content. If watermarked content is stored in a lot of popular user-generated content, Collage users can gain some level of deniability simply because all

```
send(identifier, data)
1   Create a rateless erasure encoder for data.
2   for each suitable vector (e.g., image file)
3       do
4           Retrieve blocks from the erasure coder to
                meet the vector's encoding capacity.
5           Concatenate and encrypt these blocks using
                the identifier as the encryption key.
6           encode the ciphertext into the vector.
7           Publish the vector on a user-generated
                content host such that receivers
                can find it. See Section 4.3.3.
```

```
receive(identifier)
 1  Create a rateless erasure decoder.
 2  while the decoder cannot decode the message
 3      do
 4          Find and fetch a vector from a
                user-generated content host.
 5          Check if the vector contains encoded
                data for this identifier.
 6          if the vector is encoded with message data
 7            then
 8                decode payload from the vector.
 9                Decrypt the payload.
10                Split the plaintext into blocks.
11                Provide each decrypted block to
                      the erasure decoder.
12  return decoded message from erasure decoder
```

Figure 14: The message layer's send and receive operations.

popular content contains some message chunks.

We have implemented two example vector layers. The first is image steganography applied to images hosted on Flickr [66]. The second is text steganography applied to user-generated text comments on websites such as blogs, YouTube [166], Facebook [56], and Twitter [147]. Despite possible and known limitations to these approaches (*e.g.*, [71]), both of these techniques have working implementations with running code [116, 136]. As watermarking and other data-hiding techniques continue to become more robust to attack, and as new techniques and implementations emerge, Collage's layered model can incorporate those mechanisms. The goal of this chapter is *not* to design better data-hiding techniques, but rather to build a censorship-resistant message channel that leverages these techniques.

4.3.2 Message Layer

The message layer specifies a protocol for using the vector layer to send and receive arbitrarily long messages (*i.e.*, exceeding the capacity of a single vector). Observable behavior generated by the message layer should be deniable with respect to the normal behavior of the user or users at large.

Figure 14 shows the send and receive operations. send encodes message data in vectors and publishes them on content hosts, while receive finds encoded vectors on content hosts and decodes them to recover the original message. The sender associates a message identifier with each message, which should be unique for an application (*e.g.*, the hash of the message). Receivers use this identifier to locate the message. For encoding schemes that require a key (*e.g.*, [116]), we choose the key to be the message identifier.

To distribute message data among several vectors, the protocol uses rateless erasure coding [27, 100], which generates a near-infinite supply of short chunks from a source message such that any appropriately-sized subset of those chunks can reassemble the original message. For example, a rateless erasure coder could take a 80 KB message and generate 1 KB chunks such that any 100-subset of those chunks recovers the original message. Step 1 of send initializes a rateless erasure encoder for generating chunks of the message; step 4 retrieves chunks from the encoder. Likewise, step 1 of receive creates a rateless erasure decoder, step 11 provides retrieved chunks to the decoder, and step 12 recovers the message.

Most of the remaining send operations are straightforward, involving encryption and concatenation (step 5), and operation of the vector layer's encode function (step 6). Likewise, receive operates the vector layer's decode function (step 8), decrypts and splits the payload (steps 9 and 10). The only more complex operations are step 7 of send and step 4 of receive, which publish and retrieve content from user-generated content hosts. These steps must ensure (1) that senders and receivers agree on locations of vectors and (2) that publishing and retrieving vectors is done in a deniable manner. We now describe how to meet these two requirements.

4.3.3 Rendezvous: Matching Senders to Receivers

Vectors containing message data are stored to and retrieved from user-generated content hosts; to exchange messages, senders and receivers must first *rendezvous*. To do so, senders and receivers perform sequences of *tasks*, which are time-dependent sequences of actions. An example of a sender task is the sequence of HTTP requests (*i.e.*, actions) and fetch times corresponding to "Upload photos tagged with 'flowers' to Flickr"; a corresponding receiver task is "Search Flickr for photos tagged with 'flowers' and download the first 50 images." This scheme poses many challenges: (1) to achieve deniability, all tasks must resemble observable actions completed by innocuous entities not using Collage (*e.g.*, browsing the Web), (2) senders must identify vectors suitable for each task, and (3) senders and receivers must agree on which tasks to use for each message. This section addresses these challenges.

Identifying suitable vectors. Task deniability depends on properly selecting vectors for each task. For example, for the receiver task "search for photos with keyword *flowers*," the corresponding sender task ("publish a photo with keyword *flowers*") must be used with photos of flowers; otherwise, the censor could easily identify vectors containing Collage content as those vectors that do not match their keywords. To achieve this, the sender picks vectors with attributes (*e.g.*, associated keywords) that match the expected content of the vector.

Agreeing on tasks for a message. Each user maintains a list of deniable tasks for common behaviors involving vectors (Section 4.3.1) and uses this list to construct a *task database*. The database is simply a table of pairs (T_s, T_r), where T_s is a sender task and T_r is a receiver task. Senders and receivers construct pairs such that T_s publishes vectors in locations visited by T_r. For example, if T_r performs an image search for photos with keyword "flowers" then T_s would publish only photos with that keyword (and actually depicting flowers). Given this database, the sender and receiver map each message identifier to one or more task pairs and execute T_s and T_r, respectively.

The sender and receiver must agree on the mapping of identifiers to database entries; otherwise, the receiver will be unable to find vectors published by the sender. If the sender's and receiver's databases are identical, then the sender and receiver simply use the message identifier as an index into the task database. Unfortunately, the database may change over time, for a variety of reasons: tasks become obsolete (*e.g.*, Flickr changes its page structure) and new tasks are added (*e.g.*, it may be advantageous to add a task for a new search keyword during a current event, such as an election). Each time the database changes, other users need to be made aware of these changes. To this end, Collage provides two operations on the task database: add and remove. When a user receives an advertisement for a new task or a withdrawal of an existing task he uses these operations to update his copy of the task database.

Learning task advertisements and withdrawals is application specific. For some applications, a central authority sends updates using Collage's own message layer, while in others updates are sent offline (*i.e.*, separate from Collage). We discuss these options in Section 4.5. One feature is common to all applications: delays in propagation of database

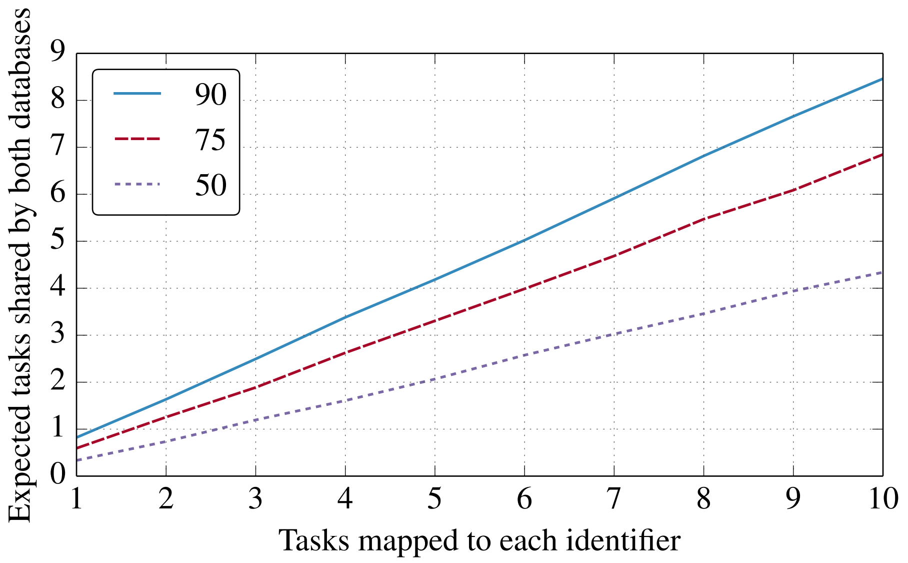


Figure 15: The expected number of common tasks when mapping the same message identifier to a task subset, between two task databases that agree on varying percentages of tasks.

updates will cause different users to have slightly different versions of the task database, necessitating a mapping for identifiers to tasks that is robust to slight changes to the database.

To reconcile database disagreements, our algorithm for mapping message identifiers to task pairs uses *consistent hash functions* [91], which guarantee that small changes to the space of output values have minimal impact on the function mapping. We initialize the task database by choosing a pseudorandom hash function h (*e.g.*, SHA-1) and precomputing $h(t)$ for each task t. The algorithm for mapping an identifier M to a m-subset of the database is simple: compute $h(M)$ and take the m entries from the task database with precomputed hash values closest to $h(M)$; these task pairs are the mapping for M.

Using consistent hashing to map identifiers to task pairs provides an important property: updating the database results in only small changes to the mappings for existing identifiers. Figure 15 shows the expected number of tasks reachable after removing a percentage of the task database and replacing it with new tasks. As expected, increasing the number of tasks mapped for each identifier decreases churn. Additionally, even if half of the database is replaced, the sender and receiver can agree on at least one task when three or more tasks are mapped to each identifier. In practice, we expect the difference between two task databases to be around 10%, so three tasks to each identifier is sufficient. Thus, two parties with slightly different versions of the task database can still communicate messages: although some tasks performed by the receiver (*i.e.*, mapped using his copy of the database) will not yield content, most tasks will.

Choosing deniable tasks. Tasks should mimic the normal behavior of users, so that a user who is performing these tasks is unlikely to be pinpointed as a Collage user (which, in and of itself, could be incriminating). We design task sequences to "match" those of normal visitors to user-generated content sites. Tasks for different content hosts have different deniability criteria. For example, the task of looking at photos corresponding to a popular tag or tag pair offers some level of deniability, because an innocuous user might be looking at popular images anyway. The challenge, of course, is finding sets of tasks that are deniable, yet focused enough to allow a user to retrieve content in a reasonable amount of time. We discuss the issue of deniability further in Section 4.6.

4.3.4 Implementation

Collage requires minimal modification to existing infrastructure, so it is small and self-contained, yet modular enough to support many possible applications; this should facilitate adoption. We have released a version of Collage [36].

We have implemented Collage as a 650-line Python library, which handles the logic of the message layer, including the task database, vector encoding and decoding, and the erasure coding algorithm. To execute tasks, the library uses Selenium [1], a popular web browser automation tool; Selenium visits web pages, fills out forms, clicks buttons and downloads vectors. Executing tasks using a real web browser frees us from implementing an HTTP client that produces realistic Web traffic (*e.g.*, by loading external images and scripts, storing cookies, and executing asynchronous JavaScript requests).

Because Collage uses Selenium to drive a real Web browser, it is the first *hide-within* censorship circumvention tool; this new class of tools tunnels its communications inside the traffic of a real, running instance of a popular application (Firefox) rather than trying to mimic that application's traffic. Hide-within circumvention tools are significantly more robust against many kinds of active and passive side-channel attacks [84]. Although Geddes *et al.* have raised concerns that hide-within systems are still vulnerable to many trivial attacks that identify architectural differences or transparently degrade performance of the covert channel, their work focuses on systems that tunnel loss-sensitive Internet traffic (*e.g.*, Tor TCP streams) inside loss-insensitive VoIP calls [73]. In contrast, Collage itself is largely immune to such attacks because its covert channel is extremely delay tolerant, while its covert traffic (*i.e.*, Web browsing) is not as tolerant such that any disruptions to the covert channel must significantly degrade performance of the cover traffic. Other attacks from Geddes *et al.* (*e.g.*, architectural mismatches and content mismatches) don't

Table 5: Examples of sender and receiver task snippets.

Content host	Sender task	Receiver task
Flickr	PublishAsUser('User', Photo, MsgData)	FindPhotosOfFlickrUser('User')
Twitter	PostTweet('Watching the Olympics', MsgData)	SearchTwitter('Olympics')

impact Collage directly, although they may apply to a specific steganographic scheme that Collage uses. In addition, conducting such attacks at line speed would be difficult because they involve decoding and analyzing compressed image data.

We represent tasks as Python functions that perform the requisite task. Table 5 shows four examples. Each application supplies definitions of operations used by the tasks (*e.g.*, `FindPhotosOfFlickrUser`). The task database is a list of tasks, sorted by their MD5 hash; to map an identifier to a set of tasks, the database finds the tasks with hashes closest to the hash of the message identifier. After mapping, receivers simply execute these tasks and decode the resulting vectors. Senders face a more difficult task: they must supply the task with a vector suitable for that task. For instance, the task "publish a photo tagged with 'flowers"' must be supplied with a photo of flowers. We delegate the task of finding vectors meeting specific requirements to a *vector provider*. The exact details differ between applications; one of our applications searches a directory of annotated photos, while another prompts the user to type a phrase containing certain words (*e.g.*, "Olympics").

4.4 Performance Evaluation

This section evaluates Collage according to the three performance metrics introduced in Section 4.2: storage overhead on content hosts, network traffic, and transfer time. We characterize Collage's performance by measuring its behavior in response to a variety of parameters. Recall that Collage (1) processes a message through an erasure coder, (2) encodes blocks inside vectors, (3) executes tasks to distribute the message vectors to content hosts, (4) retrieves some of these vectors from content hosts, and (5) decodes the message on the receiving side. Each stage can affect performance. In this section, we evaluate how each of these factors affects the performance of the message layer; Section 4.5 presents additional performance results for Collage applications using real content hosts.

- **Erasure coding** can recover an n-block message from $(1+\frac{\epsilon}{2})n$ of its coded message blocks. Collage uses $\epsilon = 0.01$, as recommended by [100], yielding an expected 0.5% increase in storage, traffic, and transfer time of a message.
- **Vector encoding** stores erasure coded blocks inside vectors. Production steganography tools achieve encoding rates of between 0.01 and 0.05, translating to between

20 and 100 factor increases in storage, traffic, and transfer time [116]. Watermarking algorithms are less efficient; we hope that innovations in information hiding can reduce this overhead.

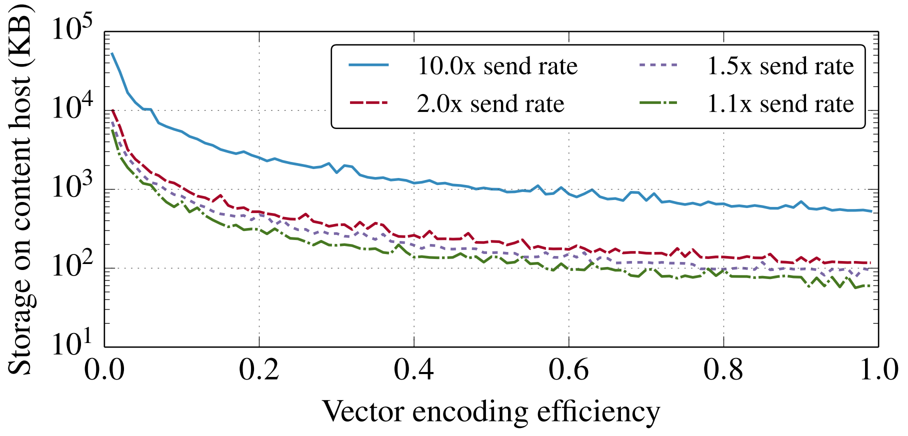
- **Sender and receiver tasks** publish and retrieve vectors from content hosts. Tasks do not affect the storage requirement on content hosts, but each task can impose additional traffic and time. For example, a task that downloads images by searching for them on Flickr can incur hundreds of kilobytes of traffic before finding encoded vectors. Depending on network connectivity, this step could take anywhere from a few seconds to a few minutes and can represent an overhead of several hundred percent, depending on the size of each vector.

- **The number of executed tasks** differs between senders and receivers. The receiver performs as many tasks as necessary until it is able to decode the message; this number depends on the size of the message, the number of vectors published by the sender, disagreements between sender and receiver task databases, the dynamics of the content host (*e.g.*, a surge of Flickr uploads could "bury" Collage encoded vectors), and the number of tasks and vectors blocked by the censor. While testing Collage, we found that we needed to execute only one task for the majority of cases.

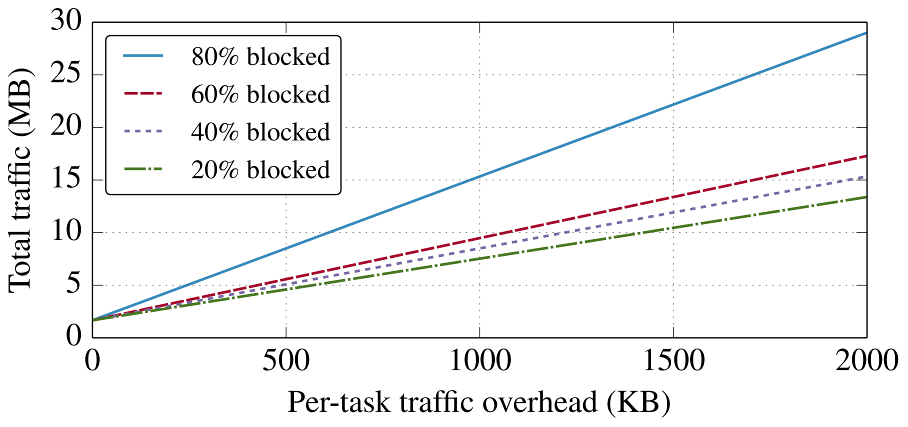
  The sender must perform as many tasks as necessary so that, given the many ways the receiver can fail to obtain vectors, the receiver will still be able to retrieve enough vectors to decode the message. In practice, this number is difficult to estimate and vectors are scarce, so the sender simply uploads as many vectors as possible.

We implemented a Collage application that publishes vectors on a simulated content host, allowing us to observe the effects of these parameters. Figure 16 shows the results of running several experiments across Collage's parameter space. The simulation sends and receives a 23 KB one-day news summary. The message is erasure coded with a block size of 8 bytes and encoded into several vectors randomly drawn from a pool for vectors with average size 200 KB. Changing the message size scales the metrics linearly, while increasing the block size only decreases erasure coding efficiency.

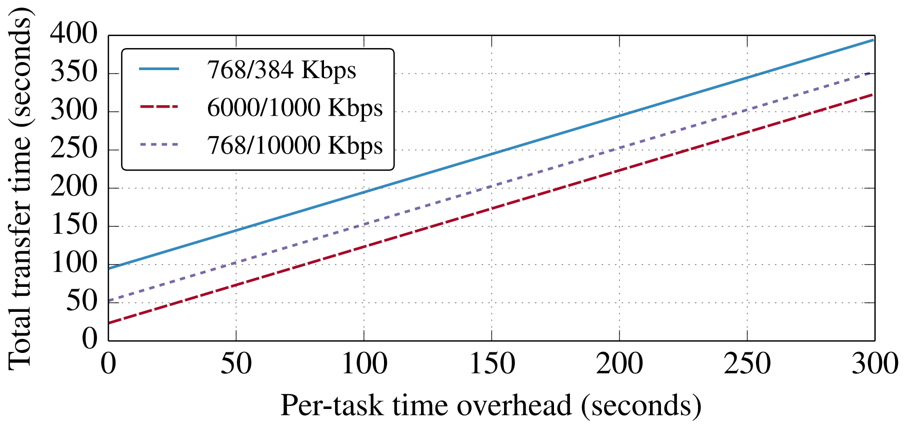
Figure 16a demonstrates the effect of vector encoding efficiency on required storage on content hosts. We used a fixed-size identifier-to-task mapping of ten tasks. We chose four *send rates*, which are multiples of the minimum number of tasks required to decode the message: the sender may elect to send more vectors if he believes some vectors may be unreachable by the receiver. For example, with a send rate of 10x, the receiver can still retrieve the message even if 90% of vectors are unavailable. Increasing the task mapping size may



(a) Storage, for various sender redundancies



(b) Traffic, for various vector block rates



(c) Transfer time, for various network connectivity rates (download/upload)

Figure 16: Collage's performance metrics, as measured using a simulated content host.

be necessary for large send rates, because sending more vectors requires executing more tasks. These results give us hope for the future of information hiding technology: current vector encoding schemes are around 5% efficient; according to Figure 16a, this a region where a significant reduction in storage is possible with only incremental improvements in encoding techniques (*i.e.*, the slope is steep).

Figure 16b predicts total sender and receiver traffic from task overhead traffic, assuming 1 MB of vector storage on the content host. As expected, blocking more vectors increases traffic, as the receiver must execute more tasks to receive the same message content. Increasing storage beyond 1 MB decreases receiver traffic, because more message vectors are available for the same blocking rate. An application executed on a real content host transfers around 1 MB of overhead traffic for a 23 KB message.

Finally, Figure 16c shows the overall transfer time for senders and receivers, given varying time overheads. These overheads are optional for both senders and receivers and impose delays between requests to evade timing analysis by the censor. For example, Collage could build a distribution of inter-request timings from the user's normal (*i.e.*, non-Collage) traffic and impose this timing distribution on Collage tasks. We simulated the total transfer time using three network connection speeds. The first (768 Kbps download and 384 Kbps upload) is a typical entry-level broadband package and would be experienced if both senders and receivers are typical users within the censored domain. The second (768/10000 Kbps) would be expected if the sender has a high-speed connection, perhaps operating as a dedicated publisher outside the censored domain; one of the applications in Section 4.5 follows this model. Finally, the 6000/1000 Kbps connection represents expected next-generation network connectivity in countries experiencing censorship. In all cases, reasonable delays are imposed upon transfers, given the expected use cases of Collage (*e.g.*, fetching daily news article). We confirmed this result: a 23 KB message stored on a real content host took under 5 minutes to receive over an unreliable broadband wireless link; sender time was less than 1 minute.

4.5 Building Applications with Collage

Developers can build a variety of applications using the Collage message channel. In this section, we outline requirements for using Collage and present two example applications.

Table 6: Summary of application components.

Component	Web content proxy (§4.5.2)	Covert email (§4.5.3)	Other options
Vectors	Photos	Text	Videos, music
Vector encoding	Image steganography	Text steganography	Video steganography, digital watermarking
Vector sources	Users of content hosts	Covert Email users	Use a heuristic, crawl the Web
Tasks	Upload/download Flickr photos	Post/receive Tweets	Other user-generated content host(s)
Database distribution	Send by publisher via proxy	Agreement by users	Prearranged algorithm, "sneakernet"
Identifier security	Distributed by publisher, groups	Group key	Existing PKI

4.5.1 Application Requirements

Even though application developers use Collage as a secure, deniable messaging primitive, they must still remain conscious of overall application security when using these primitives. Additionally, the entire vector layer and several parts of the message layer presented in Section 4.3 must be provided by the application. These components can each affect correctness, performance, and security of the entire application. In this section, we discuss each of these components. Table 6 summarizes the component choices.

Vectors, tasks, and task databases. Applications specify a class of vectors and a matching vector encoding algorithm (*e.g.*, Flickr photos with image steganography) based on their security and performance characteristics. For example, an application requiring strong content deniability for large messages could use a strong steganography algorithm to encode content inside of videos.

Tasks are application-specific: uploading photos to Flickr is different from posting tweets on Twitter. Applications insert tasks into the task database, and the message layer executes these tasks when sending and receiving messages. The applications specify how many tasks are mapped to each identifier for database lookups. In Section 4.3.3, we showed that mapping each identifier to three tasks ensures that, on average, users can still communicate even with slightly out-of-date databases; applications can further boost availability by mapping more tasks to each identifier.

Finally, applications must distribute the task database. In some instances, a central authority can send the database to application users via Collage itself. In other cases, the database is communicated offline. The application's task database should be large enough to ensure diversity of tasks for messages published at any given time; if n messages are published every day, then the database should have cn tasks, where c is at least the size of the task mapping. Often, tasks can be generated programmatically, to reduce network overhead. For example, our Web proxy (discussed next) generates tasks from a list of popular Flickr tags.

Sources of vectors. Applications must acquire vectors used to encode messages, either by requiring end-users to provide their own vectors (*e.g.*, from a personal photo collection), automatically generating them, or obtaining them from an external source (*e.g.*, a photo donation system).

Identifier security. Senders and receivers of a message must agree on a message identifier for that message. This process is analogous to key distribution. There is a general tradeoff between ease of message identifier distribution and security of the identifier: if users can easily learn identifiers, then more users will use the system, but it will also be easier for the censor to obtain the identifier; the inverse is also true. Developers must choose a distribution scheme that meets the intended use of their application. We discuss two approaches in the next two sections, although there are certainly other possibilities.

Application distribution and bootstrapping. Users ultimately need a secure one-time mechanism for obtaining the application, without using Collage. A variety of distribution mechanisms are possible: clients could receive software using spam or malware as a propagation vector, or via postal mail or person-to-person exchange. There will ultimately be many ways to distribute applications without the knowledge of the censor. Other systems face the same problem [59]. This requirement does not obviate Collage, since once the user has received the software, he or she can use it to exchange an arbitrary number of messages.

To explore these design parameters in practice, we built two applications using Collage's message layer. The first is a Web content proxy whose goal is to distribute content to many users; the second is a covert email system.

4.5.2 Web Content Proxy

We have built an asynchronous Web proxy using Collage's message layer, with which a publisher in an uncensored region makes content available to clients inside censored regimes. Unlike traditional proxies, our proxy shields both the identities of its users and the content hosted from the censor.

The proxy serves small Web documents, such as articles and blog posts, by steganographically encoding content into images hosted on photo-sharing websites like Flickr and Picasa. A standard steganography tool [116] can encode a few kilobytes in a typical image, meaning most hosted documents will fit within a few images. To host many documents simultaneously, however, the publisher needs a large supply of images; to meet this demand, the publisher operates a service allowing generous users of online image hosts to donate

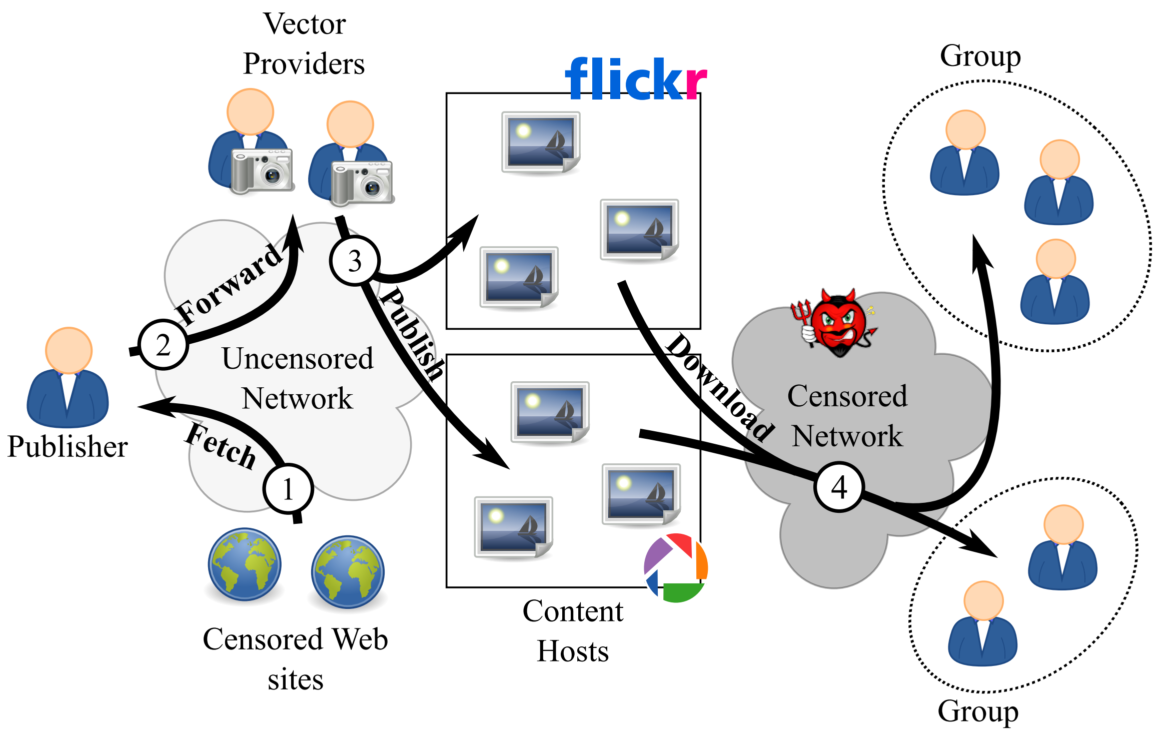


Figure 17: Proxied Web content passes through multiple parties before publication on content hosts. Each group downloads a different subset of images when fetching the same URL.

their images. The service takes the images, encodes them with message data, and returns the encoded images to their owners, who then upload them to the appropriate image hosts. Proxy users download these photos and decode their contents. Figure 17 summarizes this process. Notice that the publisher is outside the censored domain, which frees us from worrying about sender deniability.

To use a proxy, users must *discover* a publisher, *register* with that publisher, and be *notified* of an encryption key. Publishers are identified by their public key so discovering publishers is reduced to a key distribution exercise, albeit that these keys must be distributed without the suspicion of the censor. Several techniques are feasible: the key could be delivered alongside the client software, derived from a standard SSL key pair, or distributed offline. Like any key-based security system, our proxy must deal with this inherent bootstrapping problem.

Once the client knows the publisher's public key, it sends a message requesting registration. The message identifier is the publisher's public key and the message payload contains the public key of the client encrypted using the publisher's public key. This encryption ensures that only the publisher knows the client's public key. The publisher receives and decrypts the client's registration request using his own private key.

The client is now registered but doesn't know where content is located. Therefore, the publisher sends the client a message containing a *group key*, encrypted using the client's public key. The group key is shared between a small number of proxy users and is used to discover identifiers of content. For security, different groups of users fetch content from different locations; this prevents any one user from learning about (and attacking) all content available through the proxy.

After registration is complete, clients can retrieve content. To look up a URL u, a client hashes u with a keyed hash function using the group key. It uses the hash as the message identifier for receive.

Unlike traditional Web proxies, only a limited amount of content is available though our proxy. Therefore, to accommodate clients' needs for unavailable content, clients can suggest content to be published. To suggest a URL, a client sends the publisher a message containing the requested URL. If the publisher follows the suggestion, then it publishes the URL for users of that client's group key.

Along with distributing content, the publisher provides updates to the task database via the proxy itself (at the URL `proxy://updates`). The clients occasionally fetch content from this URL to keep synchronized with the publisher's task database. The consistent hashing algorithm introduced in Section 4.3.3 allows updates to be relatively infrequent; by default, the proxy client updates its database when 20% of tasks have been remapped due to churn (*i.e.*, there is a 20% reduction in the number of successful task executions). Figure 15 shows that there may be many changes to the task database before this occurs.

Implementation and Evaluation. We have implemented a simple version of the proxy and can use it to publish and retrieve documents on Flickr. The task database is a set of tasks that search for combinations (*e.g.*, "vacation" and "beach") of the 130 most popular tags. A 23 KB one-day news summary requires nine JPEG photos ($\approx$ 3 KB data per photo, plus encoding overhead) and takes approximately 1 minute to retrieve over a fast network connection; rendering web pages and large photos takes a significant fraction of this time. Note that the document is retrieved immediately after publication; performance decays slightly over time because search results are displayed in reverse chronological order. We have also implemented a photo donation service, which accepts Flickr photos from users, encodes them with censored content, and uploads them on the user's behalf. This donation service is available for download [36].

4.5.3 Covert Email

Although our Web proxy provides censored content to many users, it is susceptible to attack from the censor for precisely this reason: because no access control is performed, the censor could learn the locations of published URLs using the proxy itself and potentially mount denial-of-service attacks. To provide greater security and availability, we present Covert Email, a point-to-point messaging system built on Collage's message layer that excludes the censor from sending or receiving messages, or observing its users. This design sacrifices scalability: to meet these security requirements, all key distribution is done out of band, similar to PGP key signing.

Messages sent with Covert Email will be smaller and potentially more frequent than for the proxy, so Covert Email uses text vectors instead of image vectors. Using text also improves deniability, because receivers are inside the censored domain, and publishing a lot of text (*e.g.*, comments, tweets) is considered more deniable than many photos. Blogs, Twitter, and comment posts can all be used to store message chunks. Because Covert Email is used between a closed group of users with a smaller volume of messages, the task database is smaller and updated less often without compromising deniability. Additionally, users can supply the text vectors needed to encode content (*i.e.*, write or generate them), eliminating the need for an outside vector source. This simplifies the design.

Suppose a group of mutually trusted users wishes to communicate using Covert Email. Before doing so, it must establish a shared secret key, for deriving message identifiers for sending and receiving messages. This one-time exchange is done out-of-band; any exchange mechanism works as long as the censor is unaware that a key exchange takes place. Along with exchanging keys, the group establishes a task database. At present, a database is distributed with the application; the group can augment its task database and notify members of changes using Covert Email itself.

Once the group has established a shared key and a task database, its members can communicate. To send email to Bob, Alice generates a message identifier by encrypting a tuple of his email address and the current date, using the shared secret key. The date serves as a salt and ensures variation in message locations over time. Alice then sends her message to Bob using that identifier. Here, Bob's email address is used only to uniquely identify him within the group; in particular, the domain portion of the address serves no purpose for communication within the group.

To receive new mail, Bob attempts to receive messages with identifiers that are the encryption of his email address and some date. To check for new messages, he checks each date since the last time he checked mail. For example, if Bob last checked his mail yesterday, he checks two dates: yesterday and today.

If one group member is outside the censored domain, then *Covert Email can interface with traditional email.* This user runs an email server and acts as a proxy for the other members of the group. To send mail, group members send a message to the proxy, requesting that it be forwarded to a traditional email address. Likewise, when the proxy receives a traditional email message, it forwards it to the requisite Covert Email user. This imposes one obvious requirement on group members sending mail using the proxy: they must use email addresses where the domain portion matches the domain of the proxy email server. Because the domain serves no other purpose in Covert Email addresses, implementing this requirement is easy.

Implementation and Evaluation. We have implemented a prototype application for sending and retrieving Covert Email. Currently, the task database is a set of tasks that search posts of other Twitter users. We have also written tasks that search for popular keywords (*e.g.*, "World Cup"). To demonstrate the general approach, we have implemented an (insecure) proof-of-concept steganography algorithm that stores data by altering the capitalization of words. Sending a short 194-byte message required three tweets and took five seconds. We have shown that Covert E-mail has the potential to work in practice, although this application obviously needs many enhancements before general use, most notably a secure text vector encoding algorithm and more deniable task database.

4.6 Threats to Collage

This section discusses limitations of Collage in terms of the security threats it is likely to face from censors; we also discuss possible defenses. Recall from Section 4.2.2 that we are concerned with two security metrics: *availability* and *deniability*. Given the unknown power of the censor and lack of formal information hiding primitives in this context, both goals are necessarily best effort.

4.6.1 Availability

A censor may try to prevent clients from sending and receiving messages. Our strongest argument for Collage's availability depends on a censor's unwillingness to block large quantities of legitimate content. This section discusses additional factors that contribute to Collage's current and future availability.

The censor could *block message vectors*, but a censor that wishes to allow access to legitimate content may have trouble doing so since censored messages are encoded inside otherwise legitimate content, and message vectors are, by design, difficult to remove without destroying the cover content. Furthermore, some encoding schemes (*e.g.*, steganography) are resilient against more determined censors, because they hide the presence of Collage data; blocking encoded vectors then also requires blocking many legitimate vectors.

The censor might instead *block traffic patterns resembling Collage's tasks*. From the censor's perspective, doing so may allow legitimate users access to content as long as they do not use one of the many tasks in the task database to retrieve the content. Because tasks in the database are "popular" among innocuous users by design, blocking a task may also disrupt the activities of legitimate users. Furthermore, if applications prevent the censor from knowing the task database, mounting this attack becomes quite difficult.

The censor could *block access to content hosts*, thereby blocking access to vectors published on those hosts. Censors have mounted this attack in practice; for example, China is currently blocking Flickr and Twitter, at least in part [139]. Although Collage cannot prevent these sites from being blocked, applications can reduce the impact of this action by publishing vectors *across many user-generated content sites*, so even if the censor blocks a few popular sites there will still be plenty of sites that host message vectors. One of the strengths of Collage's design is that it does not depend on any specific user-generated content service: any site that can host content for users can act as a Collage drop site.

The censor could also try to *prevent senders from publishing content*. This action is irrelevant for applications that perform all publication outside a censored domain. For others, it is impractical for the same reasons that blocking receivers is impractical. Many content hosts (*e.g.*, Flickr, Twitter) have third-party publication tools that act as proxies to the publication mechanism [149]. Blocking all such tools is difficult, as evidenced by Iran's failed attempts to block Twitter [37].

Instead of blocking access to publication or retrieval of user-generated content, the censor could *coerce content hosts* to remove vectors or disrupt the content inside them. For certain vector encodings (*e.g.*, steganography) the content host may be unable to identify vectors containing Collage content; in other cases (*e.g.*, digital watermarking), removing encoded content is difficult without destroying the outward appearance of the vector (*e.g.*, removing the watermark could induce artifacts in a photograph).

4.6.2 Deniability

As mentioned in Section 4.2.1, the censor may try to compromise the deniability of Collage users. Intuitively, a Collage user's actions are *deniable* if the censor cannot distinguish the use of Collage from "normal" Internet activity. Deniability is difficult to quantify; others have developed metrics for anonymity [133], and we are working on quantitative metrics for deniability in our ongoing work. Instead, we explore deniability somewhat more informally and aim to understand how a censor can attack a Collage user's deniability and how future extensions to Collage might mitigate these threats. The censor may attempt to compromise the deniability of either the sender or the receiver of a message. We explore various ways the censor might mount these attacks, and possible extensions to Collage to defend against them.

The censor may attempt to **identify senders**. Applications can use several techniques to improve deniability. First, they can *choose deniable content hosts*; if a user has never visited a particular content host, it would be unwise to upload lots of content there. Second, *vectors must match tasks*; if a task requires vectors with certain properties (*e.g.*, tagged with "vacation"), vectors not meeting those requirements are not deniable. The vector provider for each application is responsible for ensuring this. Finally, *publication frequency must be indistinguishable* from a user's normal behavior and the publication frequency of innocuous users.

The censor may also attempt to **identify receivers**, by observing their task sequences. Several application parameters affect receiver deniability. As the *size of the task database* grows, clients have more variety (and thus deniability), but must crawl through more data to find message chunks. Increasing the *number of tasks mapped to each identifier* gives senders more choice of publication locations, but forces receivers to sift through more content when retrieving messages. Increasing *variety of tasks* increases deniability, but requires a human author to specify each type of task. The receiver must decide an *ordering of tasks* to visit; ideally, receivers only visit tasks that are popular among innocuous users.

Ultimately, the censor may develop more sophisticated techniques to defeat user deniability. For example, a censor may try to target individual users by mounting timing attacks (as have been mounted against other systems like Tor [7, 103]), or may look at how browsing patters change across groups of users or content sites. In these cases, we believe it is possible to extend Collage so that its request patterns more closely resemble those of innocuous users. To do so, Collage could monitor a user's normal Web traffic and allow Collage traffic to only perturb observable distributions (*e.g.*, inter-request timings, traffic per day, etc.) by small amounts. Doing so could obviously have massive a impact on Collage's performance. Preliminary analysis shows that over time this technique could yield sufficient bandwidth for productive communication, but we leave its implementation to future work.

4.7 Summary

Internet users in many countries need safe, robust mechanisms to publish content and the ability to send or publish messages in the face of censorship. Existing mechanisms for bypassing censorship firewalls typically rely on establishing and maintaining infrastructure outside the censored regime, typically in the form of proxies; unfortunately, when a censor blocks these proxies, the systems are no longer usable. This chapter presented Collage, which bypasses censorship firewalls by piggybacking messages on the vast amount and types of user-generated content on the Internet today. Collage focuses on providing both availability and some level of deniability to its users, in addition to more conventional security properties.

Collage is a further step in the ongoing arms race to circumvent censorship. As we discussed, it is likely that, upon seeing Collage, censors will take the next steps towards disrupting communications channels through the firewall—perhaps by mangling content, analyzing joint distributions of access patterns, or analyzing request timing distributions. However, as Bellovin points out: "There's no doubt that China—or any government so-minded—can censor virtually everything; it's just that the cost—cutting most communications lines, and deploying enough agents to vet the rest—is prohibitive. The more interesting question is whether or not 'enough' censorship is affordable." [14] Although Collage itself may ultimately be disrupted or blocked, it represents another step in making censorship more costly to the censors; we believe that its underpinnings—the use of user-generated content to pass messages through censorship firewalls—will survive, even as censorship techniques grow increasingly more sophisticated.

CHAPTER V

CONCLUSION

5.1 *Summary of contributions*

This thesis has demonstrated that users worldwide can assist in the measurement and circumvention of Internet censorship. In doing so, we made the following contributions:

1. *A system for measuring censorship from unmodified Web browsers.* We designed and implemented Encore, the first system that measures Internet censorship from unmodified Web browsers. Encore collects measurements of Web censorship from far more vantage points than previously possible. It is able to do so by placing deployment burden on bystanding webmasters rather than users in censored countries.

2. *A technique for circumventing censorship with user-generated content.* We explored what future censorship circumvention might look like with Collage, an automated technique for disseminating censored data by hiding it amongst legitimate content across the Web. Collage's security properties will be invaluable against future censors, which could be significantly more powerful than those today. Collage leverages content generated by real users hosted on real user-generated content hosts; using these bystanders strengthens Collage's security.

5.2 *Lessons learned*

We conclude with general lessons we heave learned since we began studying Internet censorship in 2008. In some cases, these are based on the experiences and beliefs of the author rather than claims backed by evidence.

5.2.1 For circumvention, a panacea is unlikely

As explained at the beginning of this dissertation, Internet censorship is an arms race, and one that is nowhere near resolution. New censorship and circumvention techniques will continue to arise for the considerable future. For several reasons, we believe that we may never arrive at the "perfect" circumvention tool that defeats every known censorship technique. First, as we explained in Chapter 1, Internet censorship is ultimately the result of non-technical policy; among other things, this means that subtly different censorship policies could result in massive changes in censorship mechanisms and infrastructure, perhaps employing entirely unseen techniques and requiring new circumvention techniques. Second, performance of any such panacea system could likely be so terrible as to not qualify as a panacea; we often trade off performance to gain security, and such a panacea would need to surmount a plethora of technical roadblocks. Finally, and perhaps most importantly, because censorship is simply applications of network security in new contexts and circumvention techniques represent attacks on these security systems, any technique that fundamentally broke censorship mechanisms will have profound ramifications beyond censorship and censorship circumvention; the security of the entire Internet ecosystem would be at risk.

5.2.2 We need better incentives for measurement and circumvention

Although some people genuinely want to help measure and circumvent Internet censorship, better incentives would help drive adoption of the tools we develop. Finding and convincing users around the world to install censorship measurement software is a monumental challenge; although deploying Encore has been far easier, it isn't without its own problems. Convincing webmasters to expose their users to unproven third-party measurement scripts (*i.e.*, Encore's measurement tasks) is difficult without offering something concrete in return. Similarly, convincing users to donate their photo collections would have required far more work beyond the prototype we developed and it isn't clear how many users would have used it anyway. Providing true value to users beyond simply appealing to their goodwill or curiousity could address these problems.

5.2.3 Sometimes, security isn't as important as usability

At several points during our research, we were forced to forgo perfect security for the sake of usability and practicality. We developed Encore because we lacked an easy way to collect censorship measurements from a diverse set of vantage points. Doing so required abandoning many security mechanisms present in most prior measurement collection tools. Although this could cause us to draw erroneous conclusions from measurements tampered by a censor, we realized that Encore fits into an ecosystem of censorship measurement tools and independent observers; rather than collecting ground truth, Encore's job is simply to corroborate beliefs built from data collected by other tools and assembled from people experiencing censorship first hand.

More generally, the extraordinarily high bootstrapping cost of most censorship measurement and circumvention tools means that opportunities to trade security for usability will arise more often than they otherwise might. Put another way, giving users an insecure tool is sometimes better than giving them no tool at all.

5.3 Future work

We conclude by summarizing promising research to be done in censorship discovery, measurement, and circumvention.

5.3.1 Richer censorship measurements

Although we argued in Chapter 3 that custom censorship measurement tools are extremely hard to deploy, they are probably the only way to gather rich, detailed data about Internet censorship. Even though Encore has the potential to easily collect measurements without massive deployment hurdles, there are severe and fundamental limitations on the kinds of data it can collect. Ultimately, the censorship measurement community needs to build and deploy a network of custom measurement tools — an arduous task, but necessary nonetheless. Researchers have yet to deploy and sustain a platform for measuring Internet censorship from a globally diverse set of vantage points for the long term. Luckily, we are aware of several such efforts underway that are attempting exactly this.

The author's experiences in building, deploying, and maintaining the BISmark home router testbed have shown that placing measurement infrastructure in the hands of real users introduces unexpected challenges, far beyond those faced by testbeds deployed in controlled education networks, like PlanetLab [138]. Achieving the breadth of measurements required to paint an accurate picture of Internet censorship will likely require deployment on BISmark or a platform that faces similar challenges. Collecting detailed censorship measurements also introduces new ethical and legal hurdles that researchers don't typically consider when designing general network measurement platforms [23, 162].

Figure 18 adds a new point, labeled *ICLab*, to the space of related work from Figure 4. ICLab is a project currently in development that attempts to improve on both the depth and breadth of existing censorship measurement efforts. We outline several hurdles that ICLab or similar efforts may face when attempting to deploy hardware or software in the hands of users around the world. None of these are fundamental roadblocks; rather, they are practical concerns that could make it significantly harder to sustain the measurement infrastructure. Please refer to our paper on BISmark deployment experiences for more in

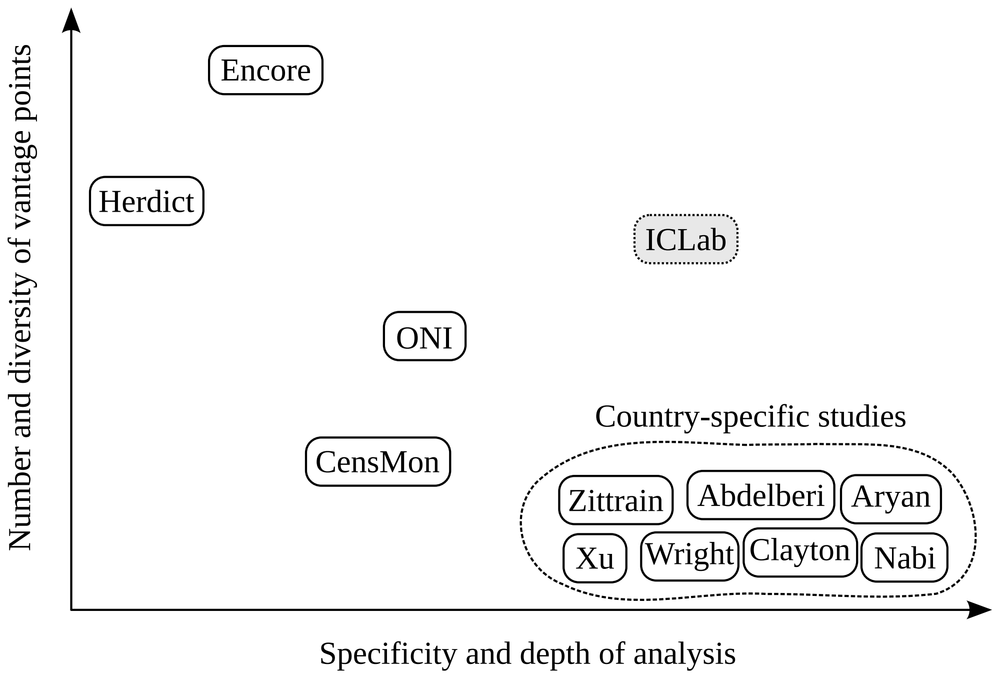


Figure 18: Future censorship measurement platforms will improve both the breadth and depth of measurements. One such effort, ICLab, is already underway.

depth discussion of several of these issues [138].

Censorship measurement is not unified. As we explained in Chapter 1, Internet censorship and Internet censorship measurement lack any fundamental technical underpinnings, which influences the design of any system that attempts to measure a wide breadth of censorship techniques. Such systems will need to touch a wide range of technologies that span the entire network stack (*e.g.*, network diagnostics like ping and traceroute; raw packet capture; deep packet inspection; TCP parameter inspection and tuning; control over packet fragmentation; fine grained manipulation of application logic) and interact with many unrelated systems (*e.g.*, arbitrary Internet server; specialized measurement servers; the plethora of middleboxes deployed by ISPs; DNS infrastructure; firewalls).

This complicates deployment because testbed authors will need to (1) design many kinds of measurements and audit their validity, security, and performance, and (2) write and test lots of code to implement those measurements. These tasks may have little overlap from one kind of measurement to another. Existing attempts to capture a wide variety of censorship measurements (*e.g.*, OONI [64]) have already realized the difficulty of capturing a breadth of censorship measurements. Such efforts typically end up shoehorning their measurements into a small set of awkward abstractions that results in implementations that are long, complicated, and ultimately unintelligible. This doesn't mean those efforts are ill-conceived, but rather illustrates the extreme difficultly of implementing censorship measurements within a single framework or set of abstractions. Rather than focusing on constructing a general framework for censorship measurement, we advise developers to focus on resource isolation and security mechanisms that will be necessary when collecting all such measurements.

Maintaining a measurement platform can be a full time job. At scale, seemingly trivial tasks require a lot of effort. For example, deploying a faulty update to a single node has only modest consequences; at worst, we lose access to a single node, and in most circumstances we can manually fix the problem remotely with a bit of effort. Deploying the same faulty update to hundreds of globally distributed nodes can have catastrophic consequences; we alienate users, must spend dozens or hundreds of hours rectifying the problem, and potentially lose access to thousands or tens of thousands of dollars worth of infrastructure. Therefore, even small upgrades require *a lot* of testing, as we must thoroughly review any source code changes and closely monitor the upgrade on a small number of nodes before widespread deployment.

Keeping measurement nodes online is an uphill battle. This is particularly true when users have different incentives and values than the developers of the measurement platform. For example, developers in countries with plentiful bandwidth and cheap power may forget that nodes could be deployed in countries with miniscule data allowances and expensive power; whereas users in some parts of world may be accustomed to leaving devices turned on all the time, users in others may view network-connected devices as appliances that should be turned off after use. Other users may be enthusiastic about leaving devices online to collect measurements in some periods of interest (*e.g.*, during an election or political unrest) but lose interest once the conflict is resolved, not realizing that continuous measurement collection is crucial for the establishment of baseline knowledge. These differences in incentives and attitudes could be a significant impediment to continuous collection of measurements.

Be wary of placing hardware on the critical path. Testbed authors may be tempted to place hardware on the critical path, either as a means of collecting more data (*e.g.*, about user activity) or to provide benefit to users (*e.g.*, by operating as a wireless access point). There are advantages and disadvantages to doing so. Advantages include the ability to observe more data about network usage and potentially compensate for measurement inaccuracies (*e.g.*, by pausing measurement collection when another client is streaming a video over the same connection), and for "free" availability monitoring (*i.e.*, users of the device will notice if it stops working). Disadvantages include users' very low tolerance for faults (in all likelihood, they will disconnect the device at the first sign of trouble) and privacy concerns of accidentally collecting personal information.

Building trusted relationships is important. Developing trusted relationships with users is critical for keeping measurement devices online. Users should feel personally invested and accountable for keeping devices online. The deployment model directly determines the establishment of these relationships. Deployment models include giving measurement devices to close friends or colleagues, or outsourcing the deployment burden to third parties in exchange for some form of compensation. For example, while deploying BISmark, we found it particularly useful to delegate deployment of nodes in foreign countries to local experts in those countries; this scales well for a few dozen nodes per delegate. On the other hand, lack of trust leads to an unstable deployment of nodes that could go offline at any time; in the extreme, we cannot trust data coming from nodes operated by arbitrary parties who may contribute malicious data (*e.g.*, if anyone can operate a measurement node, a government may elect to operate its own set of nodes to contribute false or misleading measurements).

5.3.2 Reducing bias in censorship discovery

Censorship measurement systems are ultimately only useful if we provide them with targets to measure. These inputs are provided by censorship discovery. (See Section 2.3 for a refresher.) Existing measurement tools and studies rely on manual discovery, where regional experts (*e.g.*, users affected by censorship) compile lists of censored resources to test. This means that current studies aren't measuring censorship so much as verifying cases of suspected censorship.

Rather than relying on necessarily biased human experts for censorship discovery, we could imagine building a system to produce this list automatically. Encore provides one possible avenue. With very widespread deployment (*e.g.*, installation on a very popular Web page, like *google.com*) one could imagine using Encore to test *every* known resource for censorship, regardless of suspicion. Such a deployment is exceedingly unlikely, however, and would likely be prohibitively expensive even if it happened.

An alternate approach is to shrink the size of the target list by inferring censorship from server-side usage logs. For example, we could only add resources to the target list when we detect sudden drops in traffic to them. Unfortunately, all attempts to solve the problem so far have been riddled by so many false positives as to make their output useless. This isn't for lack of effort — the author spent nearly two years experimenting with various censorship discovery techniques on three different data sets. Ultimately, this approach may work if given rich enough data about how users worldwide access Internet services.

5.3.3 Fundamental constraints on bootstrapping

Collage provides a secure yet slow covert communication channel that could serve as a rendezvous system to bootstrap faster circumvention tools like Tor. Unfortunately, Collage itself requires fairly substantial bootstrapping — users must download and install the Collage software. This bootstrapping issue affects all circumvention tools but we lack an understanding of its fundamental constraints. Some amount of bootstrapping communication must be done outside any circumvention system, but it is unclear if there are inherent limits on how small that communication can be or whether we can increase its resilience to blocking even without the help of a circumvention tool.

5.3.4 Measuring subtler forms of censorship

This dissertation only considered forms of network censorship that are easy to detect. As the arms race between censorship and circumvention advances, censors may begin employing more sophisticated and subtle forms of censorship. We already see a trend toward less invasiveness, with countries like Pakistan moving from censorship of entire domains to only certain URLs [105], and China and Iran degrading performance rather than completely disrupting communications [78, 89]. The future may blur the line between censorship and personalization by tailoring censorship for each users or modifying rather than disrupting content [118, 164].

APPENDIX A

EXAMPLE OF AN ENCORE MEASUREMENT TASK

This is a complete example of JavaScript code that runs in a client's Web browser to measure Web filtering using cross-origin embedding of a hidden image. It uses jQuery [90]. The coordination server minifies and obfuscates the source code before sending it to a client.

See `http://goo.gl/l8GU0R` for a simple demo of Encore's cross-origin request mechanism.

```
var M = Object();

// A measurement ID is a unique identifier
// linking all submissions of a measurement.
M.measurementId = ...  // a UUID.

// This function embeds an image from
// a remote origin, hides it, and
// sets up callbacks to detect success
// or failure to load the image.
M.measure = function() {
  var img = $('<img>');
  img.attr('src', '//target/image.png');
  img.style('display', 'none');
  img.on('load', M.sendSuccess);
  img.on('error', M.sendFailure);
  img.appendTo('html');
}

// This function submits a result using
// a cross-origin AJAX request. The server
// must allow such cross-origin submissions.
M.submitToCollector = function(state) {
  $.ajax({
    url: "//collector/submit" +
        "?cmh-id=" + this.measurementId +
        "&cmh-result=" + state,
  });
}
M.sendSuccess = function() {
  M.submitToCollector("success");
}
M.sendFailure = function() {
  M.submitToCollector("failure");
}

// Send a ping to the server as soon as
// the client loads the page, regardless
// of whether the measurement result. This
// indcates which clients attempted to
// run the measurement, even if they
// don’t submit a result.
M.submitToCollector("init");

// Run the measurement when the page loads.
$(M.measure);
```

REFERENCES

[1] "Selenium Web application testing system." `http://www.seleniumhq.org`.

[2] "Riaa sues computer-less family, 234 others, for file sharing." `http://arstechnica.com/old/content/2006/04/6662.ars`, Apr. 2006.

[3] Abdelberi, C., Cunche, M., Chen, T., Friedman, A., Cristofaro, E. D., and Kâafar, M. A., "Censorship in the wild: Analyzing web filtering in syria," *CoRR*, vol. abs/1402.3401, 2014.

[4] "Alexa top 500 global sites." `http://www.alexa.com/topsites`.

[5] "Anonymizer." `http://www.anonymizer.com/`.

[6] Aryan, S., Aryan, H., and Halderman, J. A., "Internet censorship in iran: A first look," in *3rd USENIX Workshop on Free and Open Communications on the Internet*, USENIX, 2013.

[7] Back, A., Möller, U., and Stiglic, A., "Traffic analysis attacks and trade-offs in anonymity providing systems," in *Proceedings of Information Hiding Workshop (IH 2001)* (Moskowitz, I. S., ed.), pp. 245–257, Springer-Verlag, LNCS 2137, Apr. 2001.

[8] "Baidu's internal monitoring and censorship document leaked." `http://chinadigitaltimes.net/2009/04/baidus-internal-monitoring-and-censorship-document-leaked`.

[9] Baliga, A., Kilian, J., and Iftode, L., "A web based covert file system," in *HOTOS'07: Proceedings of the 11th USENIX workshop on Hot topics in operating systems*, (Berkeley, CA, USA), pp. 1–6, USENIX Association, 2007.

[10] Barth, A., Caballero, J., and Song, D., "Secure content sniffing for web browsers, or how to stop papers from reviewing themselves," in *Security and Privacy, 2009 30th IEEE Symposium on*, pp. 360–371, IEEE, 2009.

[11] Bauer, K., McCoy, D., Grunwald, D., Kohno, T., and Sicker, D., "Low-resource routing attacks against tor," in *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2007)*, (Washington, DC, USA), Oct. 2007.

[12] Bavier, A., Bowman, M., Culler, D., Chun, B., Karlin, S., Muir, S., Peterson, L., Roscoe, T., Spalink, T., and Wawrzoniak, M., "Operating System Support for Planetary-Scale Network Services," in *Proc. First Symposium on Networked Systems Design and Implementation (NSDI)*, (San Francisco, CA), Mar. 2004.

[13] "Bbc - homepage." `http://www.bbc.co.uk/`.

[14] BELLOVIN, S. M., "A Matter of Cost." *New York Times* Room for Debate Blog. Can Google Beat China? `http://roomfordebate.blogs.nytimes.com/2010/01/15/can-google-beat-china/#steven`, Jan. 2010.

[15] BIN TARIQ, M., MOTIWALA, M., FEAMSTER, N., and AMMAR, M., "Detecting Network Neutrality Violations with Causal Inference," in *Proc. CoNEXT*, Dec. 2009.

[16] "BISMark: Broadband Internet Service Benchmark." `http://projectbismark.net/`.

[17] "blocked.html." `http://propakistani.pk/wp-content/uploads/2010/05/blocked.html`.

[18] "Bootstrap." `http://getbootstrap.com`.

[19] BORTZ, A. and BONEH, D., "Exposing private information by timing web applications," in *Proceedings of the 16th international conference on World Wide Web*, pp. 621–628, ACM, 2007.

[20] BOUCHER, P., SHOSTACK, A., and GOLDBERG, I., "Freedom systems 2.0 architecture," white paper, Zero Knowledge Systems, Inc., Dec. 2000.

[21] BRANIGAN, T., "Guardian blocked in China after story about leadership's offshore wealth," Jan. 2014. `http://www.theguardian.com/world/2014/jan/22/guardian-blocked-china-leaderships-offshore-wealth`.

[22] "Browser Security Handbook: Navigation and Content Inclusion Across Domains." `http://goo.gl/uMfTN5`.

[23] BURNETT, S. and FEAMSTER, N., "Making sense of Internet censorship: a new frontier for Internet measurement," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 3, pp. 84–89, 2013.

[24] BURNETT, S., FEAMSTER, N., and VEMPALA, S., "Defeating online censorship with Collage," NSDI demo session, Apr. 2009.

[25] BURNETT, S., FEAMSTER, N., and VEMPALA, S., "Chipping away at censorship firewalls with user-generated content," in *Proc. 19th USENIX Security Symposium*, (Washington, DC), Aug. 2010.

[26] BURNETT, S., FEAMSTER, N., and VEMPALA, S., "Circumventing censorship with collage," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4, pp. 471–472, 2010.

[27] BYERS, J. W., LUBY, M., MITZENMACHER, M., and REGE, A., "A digital fountain approach to reliable distribution of bulk data," in *Proc. ACM SIGCOMM*, (Vancouver, BC, Canada), pp. 56–67, Sept. 1998.

[28] "Archipelago Measurement Infrastructure." `http://www.caida.org/projects/ark/`.

[29] "The UCSD Network Telescope." `http://www.caida.org/projects/network_telescope/`.

[30] CAPPOS, J., BESCHASTNIKH, I., KRISHNAMURTHY, A., and ANDERSON, T., "Seattle: a platform for educational cloud computing," in *ACM SIGCSE Bulletin*, vol. 41, pp. 111–115, ACM, 2009.

[31] "Censorscope." `https://github.com/projectbismark/censorscope`.

[32] "China Web Sites Seeking Users' Names." `http://www.nytimes.com/2009/09/06/world/asia/06chinanet.html`, Sept. 2009.

[33] "Chinese blogger Xu Lai stabbed in Beijing bookshop." `http://www.guardian.co.uk/world/2009/feb/15/china-blogger-xu-lai-stabbed`, Feb. 2009.

[34] CLARKE, I., "A distributed decentralised information storage and retrieval system," Master's thesis, University of Edinburgh, 1999.

[35] CLAYTON, R., MURDOCH, S., and WATSON, R., "Ignoring the Great Firewall of China," in *Privacy Enhancing Technologies*, pp. 20–35, Springer, 2006.

[36] "Collage." `http://www.gtnoise.net/collage/`.

[37] "Could Iran Shut Down Twitter?." `http://futureoftheinternet.org/could-iran-shut-down-twitter`, June 2009.

[38] COWIE, J., "Egypt leaves the Internet." `http://www.renesys.com/2011/01/egypt-leaves-the-internet/#latest`, Jan. 2011.

[39] CRANDALL, J., ZINN, D., BYRD, M., BARR, E., and EAST, R., "ConceptDoppler: A Weather Tracker for Internet Censorship," in *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, (Arlington, VA), Oct. 2007.

[40] CRANDALL, J. R., CRETE-NISHIHATA, M., KNOCKEL, J., MCKUNE, S., SENFT, A., TSENG, D., and WISEMAN, G., "Chat program censorship and surveillance in china: Tracking tom-skype and sina uc," *First Monday*, vol. 18, no. 7, 2013.

[41] DAINOTTI, A., SQUARCELLA, C., ABEN, E., CLAFFY, K. C., CHIESA, M., RUSSO, M., and PESCAPÉ, A., "Analysis of country-wide internet outages caused by censorship," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pp. 1–18, ACM, 2011.

[42] DALEK, J., HASELTON, B., NOMAN, H., SENFT, A., CRETE-NISHIHATA, M., GILL, P., and DEIBERT, R. J., "A method for identifying and confirming the use of url filtering products for censorship," in *Proceedings of the 2013 conference on Internet measurement conference*, pp. 23–30, ACM, 2013.

[43] DANEZIS, G., "Covert communications despite traffic data retention."

[44] DANEZIS, G., "An anomaly-based censorship detection system for Tor." `https://metrics.torproject.org/papers/detector-2011-09-09.pdf`, Sept. 2011.

[45] DANEZIS, G. and DIAZ, C., "A survey of anonymous communication channels," Tech. Rep. MSR-TR-2008-35, Microsoft Research, Jan. 2008.

[46] DANEZIS, G., DINGLEDINE, R., and MATHEWSON, N., "Mixminion: Design of a Type III Anonymous Remailer Protocol," in *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, pp. 2–15, May 2003.

[47] DEIBERT, R., *Access denied: The practice and policy of global internet filtering*. MIT Press, 2008.

[48] DEIBERT, R., *Access controlled: The shaping of power, rights, and rule in cyberspace*. The MIT Press, 2010.

[49] DHAWAN, M., SAMUEL, J., TEIXEIRA, R., KREIBICH, C., ALLMAN, M., WEAVER, N., and PAXSON, V., "Fathom: a browser-based network measurement platform," in *Proceedings of the 2012 ACM conference on Internet measurement conference*, pp. 73–86, ACM, 2012.

[50] DINGLEDINE, R. and MATHEWSON, N., "Design of a blocking-resistant anonymity system," tech. rep., The Tor Project, 2006. `https://svn.torproject.org/svn/projects/design-paper/blocking.pdf`.

[51] DINGLEDINE, R., MATHEWSON, N., and SYVERSON, P., "Tor: The second-generation onion router," in *Proc. 13th USENIX Security Symposium*, (San Diego, CA), Aug. 2004.

[52] DUAN, H., WEAVER, N., ZHAO, Z., HU, M., LIANG, J., JIANG, J., LI, K., and PAXSON, V., "Hold-On: Protecting Against On-Path DNS Poisoning," in *Securing and Trusting Internet Names*, (Teddington, UK), National Physical Laboratory, 2012. `http://conferences.npl.co.uk/satin/papers/satin2012-Duan.pdf`.

[53] DYER, K. P., COULL, S. E., RISTENPART, T., and SHRIMPTON, T., "Protocol misidentification made easy with format-transforming encryption," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 61–72, ACM, 2013.

[54] Edward Wong, “Bloomberg News Is Said to Curb Articles That Might Anger China.” http://www.nytimes.com/2013/11/09/world/asia/bloomberg-news-is-said-to-curb-articles-that-might-anger-china.html, Nov. 2013.

[55] “gtnoise/encore.” https://github.com/gtnoise/encore.

[56] “Facebook.” http://www.facebook.com/.

[57] Faris, R., Roberts, H., and Wang, S., “Chinas green dam: The implications of government control encroaching on the home pc,” *OpenNet Initiative Bulletin*, 2009. http://opennet.net/sites/opennet.net/files/GreenDam_bulletin.pdf.

[58] “Fathom API.” http://fathom.icsi.berkeley.edu/docs/.

[59] Feamster, N., Balazinska, M., Harfst, G., Balakrishnan, H., and Karger, D., “Infranet: Circumventing Web censorship and surveillance,” in *Proc. 11th USENIX Security Symposium*, (San Francisco, CA), Aug. 2002.

[60] Feamster, N., Balazinska, M., Wang, W., Balakrishnan, H., and Karger, D., “Thwarting Web censorship with untrusted messenger discovery,” in *3rd Workshop on Privacy Enhancing Technologies*, (Dresden, Germany), Mar. 2003.

[61] Feamster, N. and Dingledine, R., “Location diversity in anonymity networks,” in *ACM Workshop on Privacy in the Electronic Society*, (Washington, DC), Oct. 2004.

[62] Fifield, D., Hardison, N., Ellithorpe, J., Stark, E., Boneh, D., Dingledine, R., and Porras, P., “Evading censorship with browser-based proxies,” in *Privacy Enhancing Technologies*, pp. 239–258, Springer, 2012.

[63] Fifield, D., Nakibly, G., and Boneh, D., “OSS: Using Online Scanning Services for Censorship Circumvention,” in *Privacy Enhancing Technologies Symposium*, (Bloomington, IN, USA), Springer, 2013. http://freehaven.net/anonbib/papers/pets2013/paper_29.pdf.

[64] Filastò, A. and Appelbaum, J., “Ooni: Open observatory of network interference,” in *USENIX FOCI*, Aug. 2012.

[65] “Filbaan.” http://filbaan.net.

[66] “Flickr.” http://www.flickr.com/.

[67] Freedman, M. J. and Morris, R., “Tarzan: A peer-to-peer anonymizing network layer,” in *Proc. 9th ACM Conference on Computer and Communications Security*, (Washington, D.C.), Nov. 2002.

[68] "Freedom on the Net," tech. rep., Freedom House, Mar. 2009. http://www.freedomhouse.org/uploads/specialreports/NetFreedom2009/FreedomOnTheNet_FullReport.pdf.

[69] "Freedom on the Net," tech. rep., Freedom House, Oct. 2013. http://freedomhouse.org/sites/default/files/resources/FOTN%202013_Full%20Report_0.pdf.

[70] "Freegate." http://www.dit-inc.us/freegate.

[71] FRIDRICH, J., GOLJAN, M., and HOGEA, D., "Attacking the outguess," in *Proceedings of the ACM Workshop on Multimedia and Security*, 2002.

[72] "Future of Open Source: Collaborative Culture." http://www.wired.com/dualperspectives/article/news/2009/06/dp_opensource_wired0616, June 2009.

[73] GEDDES, J., SCHUCHARD, M., and HOPPER, N., "Cover Your ACKs: Pitfalls of Covert Channel Censorship Circumvention," in *Computer and Communications Security*, (Berlin, Germany), ACM, 2013. http://www-users.cs.umn.edu/~hopper/ccs13-cya.pdf.

[74] GILL, P., CRETE-NISHIHATA, M., DALEK, J., GOLDBERG, S., SENFT, A., and WISEMAN, G., "Characterizing censorship of web content worldwide." http://www.cs.stonybrook.edu/~phillipa/papers/ONIAnaly.html.

[75] "Google censors itself for China." http://news.bbc.co.uk/2/hi/technology/4645596.stm.

[76] "Google Transparency Report." http://www.google.com/transparencyreport/.

[77] "Greatfire.org: Online censorship in china." http://en.greatfire.org/.

[78] "Gmail now 45 times slower than QQ, 8 times slower than Yahoo." https://en.greatfire.org/blog/2011/mar/gmail-now-45-times-slower-qq-8-times-slower-yahoo, Mar. 2011.

[79] GUMMADI, K. P., SAROIU, S., and GRIBBLE, S. D., "King: Estimating latency between arbitrary internet end hosts," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pp. 5–18, ACM, 2002.

[80] "HAR 1.2 spec." http://www.softwareishard.com/blog/har-12-spec/.

[81] HARDY, S., "Asia chats: Investigating regionally-based keyword censorship in line." https://citizenlab.org/2013/11/asia-chats, Nov 2013.

[82] "HerdictWeb: The Verdict of the Herd." `http://herdict.org`.

[83] "Herdict: Browse Lists." `http://herdict.org/lists`. Visited 2014-02-26.

[84] HOUMANSADR, A., BRUBAKER, C., and SHMATIKOV, V., "The parrot is dead: Observing unobservable network communications," in *Security and Privacy (SP), 2013 IEEE Symposium on*, pp. 65–79, IEEE, 2013.

[85] HOUMANSADR, A., NGUYEN, G. T. K., CAESAR, M., and BORISOV, N., "Cirripede: Circumvention infrastructure using router redirection with plausible deniability," in *Proc. ACM CCS*, Oct. 2011.

[86] HOUMANSADR, A., RIEDL, T., BORISOV, N., and SINGER, A., "I want my voice to be heard: Ip over voice-over-ip for unobservable censorship circumvention," in *The 20th Annual Network and Distributed System Security Symposium (NDSS)*, 2013.

[87] HOWARD, F., "Malware with your mocha: Obfuscation and antiemulation tricks in malicious javascript," *Sophos Technical Papers*, 2010.

[88] HUANG, L.-S., WEINBERG, Z., EVANS, C., and JACKSON, C., "Protecting browsers from cross-origin css attacks," in *Proceedings of the 17th ACM conference on Computer and communications security*, pp. 619–629, ACM, 2010.

[89] "Iran Cracks Down on Internet Use, Foreign Media." `http://online.wsj.com/news/articles/SB124519888117821213`, June 2009.

[90] "jQuery." `http://jquery.com`.

[91] KARGER, D., LEHMAN, E., LEIGHTON, T., PANIGRAHY, R., LEVINE, M., and LEWIN, D., "Consistent hashing and random trees: distributed caching protocols for relieving hot spots on the world wide web," in *STOC '97: Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, (New York, NY, USA), pp. 654–663, ACM, 1997.

[92] KARIR, M., HUSTON, G., MICHAELSON, G., and BAILEY, M., "Understanding ipv6 populations in the wild," in *Passive and Active Measurement*, pp. 256–259, Springer, 2013.

[93] KARLIN, J., ELLARD, D., JACKSON, A. W., JONES, C. E., LAUER, G., MANKINS, D. P., and STRAYER, W. T., "Decoy Routing: Toward Unblockable Internet Communication," in *USENIX Workshop on Free and Open Communications on the Internet*, (San Francisco, CA), Aug. 2011.

[94] KATHURIA, K., "Bypassing internet censorship for news broadcasters," in *FOCI11 USENIX Workshop on Free and Open Communications on the Interet*, pp. 1–6, 2011.

[95] KIRKLAND, A., "Five activists punished by their governments for speaking out." `http://www.indexoncensorship.org/2014/01/five-activists-punished-governments-speaking/`, Jan. 2014.

[96] KREIBICH, C., WEAVER, N., NECHAEV, B., and PAXSON, V., "Netalyzr: illuminating the edge network," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, 2010.

[97] LINCOLN, P., MASON, I., PORRAS, P., YEGNESWARAN, V., WEINBERG, Z., MASSAR, J., SIMPSON, W., VIXIE, P., and BONEH, D., "Bootstrapping Communications into an Anti-Censorship System," in *Free and Open Communications on the Internet*, (Bellevue, WA, USA), USENIX Association, 2012. `https://www.usenix.org/system/files/conference/foci12/foci12-final7.pdf`.

[98] MADHYASTHA, H. V., ISDAL, T., PIATEK, M., DIXON, C., ANDERSON, T. E., KRISHNAMURTHY, A., and VENKATARAMANI, A., "iPlane: An information plane for distributed services," in *Proc. 7th USENIX OSDI*, (Seattle, WA), Nov. 2006.

[99] "MaxMind GeoIP Country." `http://www.maxmind.com/app/geolitecountry`. Retrieved: June 2011.

[100] MAYMOUNKOV, P., "Online codes," Technical Report TR2002-833, New York University, Nov. 2002.

[101] MCCOY, D., MORALES, J. A., and LEVCHENKO, K., "Proximax: A Measurement Based System for Proxies Dissemination," in *Financial Cryptography and Data Security*, (St. Lucia), Springer, 2011. `http://cseweb.ucsd.edu/~klevchen/mml-fc11.pdf`.

[102] MOHAJERI MOGHADDAM, H., LI, B., DERAKHSHANI, M., and GOLDBERG, I., "Skypemorph: Protocol obfuscation for tor bridges," in *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 97–108, ACM, 2012.

[103] MURDOCH, S. J. and DANEZIS, G., "Low-cost traffic analysis of Tor," in *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, IEEE CS, May 2005.

[104] "My Speed Test." `https://play.google.com/store/apps/details?id=com.num`.

[105] NABI, Z., "The anatomy of web censorship in pakistan," in *Proceedings of the 3rd USENIX Workshop on Free and Open Communications on the Internet (FOCI13), Washington, DC. Retrieved from http://arxiv. org/pdf/1307.1144. pdf*, 2013.

[106] "Cisco NetFlow." `http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html`, 2013.

[107] "Net neutrality monitor." `http://www.neumon.org/`.

[108] "Noction: Network Intelligence." `http://www.noction.com`.

[109] "Tor project: obfsproxy." https://www.torproject.org/projects/obfsproxy.html.en.

[110] OF OREGON, U., "RouteViews." http://www.routeviews.org/.

[111] "OpenNet Initiative." http://www.opennet.net/.

[112] "OpenNet Initiative Research Publications." http://www.opennet.net/research/.

[113] "Report on china's filtering practices," 2008. Open Net Initiative. http://opennet.net/sites/opennet.net/files/china.pdf.

[114] "Open observatory of network interference." https://ooni.torproject.org.

[115] "OpenWrt." https://openwrt.org, Sept. 2013.

[116] "Outguess." http://www.outguess.org/.

[117] "Pakistan blocks YouTube over anti-Islam video," 2012. http://www.aljazeera.com/news/asia/2012/09/2012917112431580115.html.

[118] PARISER, E., *The Filter Bubble:What the Internet Is Hiding from You.* Penguin, 2011.

[119] PARK, J. C. and CRANDALL, J. R., "Empirical study of a national-scale distributed intrusion detection system: Backbone-level filtering of html responses in china," in *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on*, pp. 315–326, IEEE, 2010.

[120] PETERSON, L., BAVIER, A., FIUCZYNSKI, M. E., and MUIR, S., "Experiences building PlanetLab," in *Proceedings of the 7th symposium on Operating systems design and implementation*, pp. 351–366, USENIX Association, 2006.

[121] "Phantomjs." http://phantomjs.org.

[122] "PlanetLab." http://www.planet-lab.org/, 2010.

[123] "Psiphon — total delivery solution for the censored internet." https://psiphon.ca.

[124] REITER, M. and RUBIN, A., "Crowds: Anonymity for web transactions," *ACM Transactions on Information and System Security (TISSEC)*, vol. 1, pp. 66–92, Nov. 1998. http://www.research.att.com/projects/crowds/.

[125] "Repyapi." https://seattle.poly.edu/wiki/RepyApi.

[126] "RIPE Atlas." https://atlas.ripe.net.

[127] "RIPE Atlas – API documentation." https://atlas.ripe.net/docs/api.

[128] "Réseaux IP Européens Next Section Routing Information Service (RIS)." http://www.ripe.net/ris/.

[129] "Internet - reporters without borders," Apr. 2014. https://en.rsf.org/internet.html.

[130] "Same Origin Policy." https://developer.mozilla.org/en-US/docs/Web/JavaScript/Same_origin_policy_for_JavaScript. Mozilla Developer Network.

[131] SÁNCHEZ, M. A., OTTO, J. S., BISCHOF, Z. S., CHOFFNES, D. R., BUSTAMANTE, F. E., KRISHNAMURTHY, B., and WILLINGER, W., "Dasu: Pushing experiments to the internets edge," in *Proc. of USENIX NSDI*, 2013.

[132] "Seattle." https://seattle.poly.edu.

[133] SERJANTOV, A. and DANEZIS, G., "Towards an information theoretic metric for anonymity," in *Proceedings of Privacy Enhancing Technologies Workshop (PET 2002)* (DINGLEDINE, R. and SYVERSON, P., eds.), Springer-Verlag, LNCS 2482, Apr. 2002.

[134] SERJANTOV, A. and SEWELL, P., "Passive attack analysis for connection-based anonymity systems," in *Proceedings of ESORICS 2003*, Oct. 2003.

[135] SFAKIANAKIS, A., ATHANASOPOULOS, E., and IOANNIDIS, S., "Censmon: A web censorship monitor," in *USENIX FOCI*, Aug. 2011.

[136] "The SNOW Home Page." http://www.darkside.com.au/snow/.

[137] SOVRAN, Y., LI, J., and SUBRAMANIAN, L., "Pass it on: Social networks stymie censors," in *Proceedings of the 7th International Workshop on Peer-to-Peer Systems*, Feb. 2008.

[138] SUNDARESAN, S., BURNETT, S., FEAMSTER, N., and DE DONATO, W., "Bismark: A testbed for deploying measurements and applications in broadband access networks," in *Proceedings of the USENIX Annual Technical Conference*, 2014.

[139] TECHCRUNCH, "China Blocks Access To Twitter, Facebook After Riots." http://www.techcrunch.com/2009/07/07/china-blocks-access-to-twitter-facebook-after-riots/.

[140] "Tor: Bridges." http://www.torproject.org/bridges.

[141] "Tor metrics portal." https://metrics.torproject.org.

[142] "Tor partially blocked in China," Sept. 2009. https://blog.torproject.org/blog/tor-partially-blocked-china.

[143] "An update on the censorship in Ethiopia," June 2012. `https://blog.torproject.org/blog/update-censorship-ethiopia`.

[144] TORRENTFREAK, "China Hijacks Popular BitTorrent Sites." `http://torrentfreak.com/china-hijacks-popular-bittorrent-sites-081108/`, May 2008.

[145] "Turkey blocks YouTube days after Twitter crackdown," Mar. 2014. `http://www.cnn.com/2014/03/27/world/europe/turkey-youtube-blocked/`.

[146] "Twitter website 'blocked' in Turkey." `http://www.bbc.com/news/world-europe-26677134`, Mar. 2014.

[147] "Twitter." `http://www.twitter.com`.

[148] "18 Million Twitter Users by End of 2009." `http://mashable.com/2009/09/14/twitter-2009-stats/`, Sept. 2009.

[149] "Ultimate List of Twitter Applications." `http://techie-buzz.com/twitter/ultimate-list-of-twitter-applications-and-websites.html`, 2009.

[150] "State of the Twittersphere." `http://bit.ly/sotwitter`, 2009.

[151] "Ultrasurf — free proxy-based Internet privacy and security tools." `http://www.ultrasurf.us/`.

[152] "uProxy." `http://www.google.com/ideas/projects/uproxy/`.

[153] VERKAMP, J.-P. and GUPTA, M., "Inferring mechanics of Web censorship around the world," in *Proceedings of the 2nd USENIX Workshop on Free and Open Communications on the Internet*, Aug. 2013.

[154] WALDMAN, M. and MAZIÈRES, D., "Tangler: A censorship-resistant publishing system based on document entanglements," in *Proc. 8th ACM Conference on Computer and Communications Security*, (Philadelphia, PA), Nov. 2001.

[155] WANG, Q., LIN, Z., BORISOV, N., and HOPPER, N. J., "rBridge: User Reputation based Tor Bridge Distribution with Privacy Preservation," in *Network and Distributed System Security*, (San Diego, CA, USA), The Internet Society, 2013. `http://www-users.cs.umn.edu/~hopper/rbridge_ndss13.pdf`.

[156] "Encore Web censorship testbed." `https://github.com/sburnett/web-censorship-testbed`.

[157] WEINBERG, Z., WANG, J., YEGNESWARAN, V., BRIESEMEISTER, L., CHEUNG, S., WANG, F., and BONEH, D., "Stegotorus: a camouflage proxy for the tor anonymity system," in *Proceedings of the 2012 ACM conference on Computer and communications security*, pp. 109–120, ACM, 2012.

[158] WINTER, P., "Towards a Censorship Analyser for Tor," in *Proceedings of the 2nd USENIX Workshop on Free and Open Communications on the Internet*, 2013.

[159] WINTER, P. and LINDSKOG, S., "How the great firewall of china is blocking tor," in *Proceedings of the 2nd USENIX Workshop on Free and Open Communications on the Internet*, 2012.

[160] WINTER, P., PULLS, T., and FUSS, J., "ScrambleSuit: A Polymorphic Network Protocol to Circumvent Censorship," in *Workshop on Privacy in the Electronic Society*, ACM, 2013. URL: `http://www.cs.kau.se/philwint/pdf/wpes2013.pdf`.

[161] WRIGHT, J., "Regional Variation in Chinese Internet Filtering," tech. rep., University of Oxford, 2012. `http://papers.ssrn.com/sol3/Delivery.cfm/SSRN_ID2265775_code1448244.pdf?abstractid=2265775&mirid=3`.

[162] WRIGHT, J. and BROWN, I., "Fine-grained censorship mapping: Information sources, legality and ethics," in *USENIX FOCI*, Aug. 2011.

[163] WUSTROW, E., WOLCHOK, S., GOLDBERG, I., and HALDERMAN, J. A., "Telex: Anticensorship in the Network Infrastructure," in *Proc. 20th USENIX Security Symposium*, (San Francisco, CA), Aug. 2011.

[164] XING, X., MENG, W., DOOZAN, D., FEAMSTER, N., LEE, W., and SNOEREN, A. C., "Exposing inconsistent web search results with bobble," in *Passive & Active Measurement (PAM)*, (Los Angeles, CA), Mar. 2014.

[165] XU, X., MAO, Z. M., and HALDERMAN, J. A., "Internet censorship in China: Where does the filtering occur?," in *Passive and Active Measurement*, pp. 133–142, Springer, 2011.

[166] "Youtube - broadcast yourself." `http://www.youtube.com/`.

[167] "YouTube Statistics." `http://ksudigg.wetpaint.com/page/YouTube+Statistics`, Mar. 2008.

[168] ZITTRAIN, J. and EDELMAN, B., "Internet filtering in China," *IEEE Internet Computing*, vol. 7, no. 2, pp. 70–77, 2003.

Empowering bystanders to facilitate Internet censorship measurement and circumvention

Samuel Read Burnett

96 Pages

Directed by Professor Nick Feamster

Free and open exchange of information on the Internet is at risk: more than 60 countries practice some form of Internet censorship, and both the number of countries practicing censorship and the proportion of Internet users who are subject to it are on the rise. Understanding and mitigating these threats to Internet freedom is a continuous technological arms race with many of the most influential governments and corporations.

By its very nature, Internet censorship varies drastically from region to region, which has impeded nearly all efforts to observe and fight it on a global scale. Researchers and developers in one country may find it very difficult to study censorship in another; this is particularly true for those in North America and Europe attempting to study notoriously pervasive censorship in Asia and the Middle East.

This dissertation develops techniques and systems that empower users in one country, or *bystanders*, to assist in the measurement and circumvention of Internet censorship in another. Our work builds from the observation that there are people *everywhere* who are willing to help us if only they knew how. First, we develop Encore, which allows webmasters to help study Web censorship by collecting measurements from their sites' visitors. Encore leverages weaknesses in cross-origin security policy to collect measurements from a far more diverse set of vantage points than previously possible. Second, we build Collage, a technique that uses the pervasiveness and scalability of user-generated content to disseminate censored content. Collage's novel communication model is robust against censorship that is significantly more powerful than governments use today. Together, Encore and Collage help people everywhere study and circumvent Internet censorship.