Enhancing System Transparency, Trust, and Privacy with Internet Measurement

by

Benjamin VanderSloot

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in the University of Michigan
2020

Doctoral Committee:

Professor J. Alex Halderman, Chair
Assistant Professor Roya Ensafi
Research Professor Peter Honeyman
Assistant Professor Baris Kasikci
Professor Kentaro Toyama

Benjamin VanderSloot

admin@benvds.com

ORCID iD: 0000-0001-6528-3927

ACKNOWLEDGEMENTS

First and foremost, I thank my wife-to-be, Alexandra. Without her unwavering love and support, this dissertation would have remained unfinished. Without her help, the ideas here would not be as strong. Without her, I would not be here.

Thanks are also due to my advisor, J. Alex Halderman, for giving me the independence and support I needed to find myself. I also thank Peter Honeyman for sharing his experience freely, both with history lessons and mentorship. Thank you Roya Ensafi, for watching out for me since we met. Thank you Baris Kasikci and Kentaro Toyama, for helping me out the door, in spite of the ongoing pandemic. I owe a great debt to my parents, Craig and Teresa VanderSloot, for raising me so well. No doubt the lessons about perseverance and hard work were put to good use. Your pride has been fuel to my fire for a very long time now. I also owe so much to my sister, Christine Kirbitz, for being the best example a little brother could hope for. I always had a great path to follow and an ear to share my problems with. Matt Bernhard and Allison McDonald, I owe you my sanity. Let me know if you need to borrow it; I'll leave the door open so you can follow me out quickly. Thank you Drew Springall and David Adrian for always taking the time out of your own journey to help me. My time in grad school would not have been the same without the fun and academic challenge I had in lab, and for that I have so many to thank: Eric Wustrow, James Kasten, Zakir Durumeric, Will Scott, Chris Dzombak, Ariana Mirian, Ben Burgess, Alex Holland, Gabrielle Beck, Rose Howell, Sai Gouravajhala, Ofir Weisse, Marina Minkin, Andrew Kwong, and many others. Thank you to all of the professors that showed me kindness, helped me, or inspired me.

The work in this dissertation was supported in part by the National Science Foundation; the United States Department of State Bureau of Democracy, Human Rights, and Labor; the Rackham Merit Fellowship; and the Computer Science and Engineering Division at Michigan.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

ABSTRACT

While on the Internet, users participate in many systems designed to protect their information's security. Protection of the user's information can depend on several technical properties, including transparency, trust, and privacy. Preserving these properties is challenging due to the scale and distributed nature of the Internet; no single actor has control over these features. Instead, the systems are designed to provide them, even in the face of attackers. However, it is possible to utilize Internet measurement to better defend transparency, trust, and privacy. Internet measurement allows observation of many behaviors of distributed, Internet-connected systems. These new observations can be used to better defend the system they measure.

In this dissertation, I explore four contexts in which Internet measurement can be used to the aid of end-users in Internet-centric, adversarial settings. First, I improve transparency into Internet censorship practices by developing new Internet measurement techniques. Then, I use Internet measurement to enable the deployment of end-to-middle censorship circumvention techniques to a half-million users. Next, I evaluate transparency and improve trust in the Web public-key infrastructure by combining Internet measurement techniques and using them to augment core components of the Web public-key infrastructure. Finally, I evaluate browser extensions that provide privacy to users on the web, providing insight for designers and simple recommendations for end-users.

Thesis Statement. By focusing on end-user concerns in widely deployed systems critical to end-user security and privacy, Internet measurement enables improvements to transparency, trust, and privacy.

CHAPTER I

Introduction

With the rapid adoption and spread of the Internet, computers and the Internet have become indispensable to society. In many parts of the world, very little business is done without some use of the Internet. The Internet contains an expansive collection of human knowledge and enables communication that was unthinkable a few decades ago. Social networks play a role in how people interact with each other. The systems that constitute the Internet as we know it are massive and critical to daily life.

Unfortunately, large systems that do not consider a person's agency can treat that person poorly. Perhaps unsurprisingly due to the importance of the Internet, this can be a challenge for systems built on the Internet. Users are tracked and have personal information inferred. Often this happens without asking the user, and when the user is asked, the models of consent do not hold up to scrutiny [197]. What information users access is shaped by these inferences and governments in unaccountable ways.

Three of the technical properties that enable user agency are transparency, trust, and privacy. Transparency allows the user to observe the actions of the system so that they may understand the impacts of their behavior. Trust allows the user to control the parties that have a privileged position in acting as their agent or providing them information. Privacy allows the user to make choices freely and can be defined as a choice of when to be observed. Each of these technical properties enable user decision making.

When a system is Internet-based, several concerns are amplified. The scope of the system can grow almost arbitrarily, and beyond the intent of its creator [28]. Whether a system is designed to help or harm particular users is not as important as its emergent behavior. In this way, Internet measurement is a singularly useful tool to gain information about Internet-based systems; it analyzes precisely what the system is currently doing. This information can be shared directly, used to assert axioms of the system's design, or help users blend into a crowd.

In this dissertation, I explore the use of Internet measurement for system transparency, trust, and privacy through four case studies. First, I explore the application of Internet measurement to Internet censorship to provide transparency to a deliberately opaque practice. Second, I further explore Internet censorship and the use of Internet measurement in aiding censorship circumvention. Third, I apply Internet measurement to the evaluation of Web tracker-blocking to produce easily digestible recommendations and advice for privacy-enhancing tool designers. Fourth, I use multiple perspectives of Internet measurement to illuminate the extent of transparency in HTTPS certificates, improving transparency for the average website administrator and providing insight for improving Internet measurement.

1.1 Measuring Internet Censorship

Internet censorship is a phenomenon that has been studied from multiple perspectives, technical and non-technical, in recent years [6,14,15,35,38,41,44,46,78,113,120,132,156,157,193,208,222]. At its core, a user attempts to use services outside of its network that the network operator does not want to be used. This phenomenon is often discussed in the more specific context of citizens attempting to visit websites or parts of the Internet that the government forbids. Governments don't disclose what they are blocking, and often provide only vague descriptions of what is being censored [90]. This leads to self-censorship and an inability for users to engage in an informed discussion on the policy.

In recent years, measurement of censorship has developed techniques that no longer rely on an extensive network of in-country volunteers to maintain censorship probes [75]. Rather,

researchers developed techniques that make use of existing network infrastructure to measure censorship [69,161,162,180]. I identified a simple technique to measure commonly used application-layer censorship that takes advantage of legacy services online on the Internet today. This technique, combined with simple heuristics, is able to identify national-level censorship in practice.

This study included the largest test list of web sites for the purpose of detecting censorship, testing 100,000 web sites across 40 countries. Additionally, we tested a smaller list of sensitive domains across 82 countries, and were able to quantify and provide qualitative descriptions of censorship to support ONI and Freedom House reports. This technique, and its evolution in future work, provide one of the key technical aspects of Censored Planet, a longitudinal censorship measurement platform. Censored Planet, and the transparency it provides, were important in the discussion surrounding Kazakhstan's recent deployment of censorship tools. Users and stakeholders in the system alike can simply look at a list of blocked domains retrieved using these tools, rather than trusting the censoring countries' claims about what they are censoring and why.

1.2 Measuring to Circumvent Censorship

Deployment of censorship circumvention tools can benefit from Internet measurement techniques. At time of writing, the first continuous deployment of Refraction Networking, a next generation censorship circumvention technique, is available and being used by users in censored countries. In order to deploy this technique, a key step was to discover which websites may be used to provide plausible deniability for users in each country based upon our deployment and network conditions. To provide this list, we must understand which extant websites place our deployed network appliance on path between a typical user in a censoring country and an uncensored website. In order to do this we use Internet measurement.

We tested a deployment in the spring of 2017 on a trial basis, which I present the results of. Over a single week, 50,000 unique daily users were able to access the Internet through our deployment, which has translated to over 100,000 daily users in our current deployment. This led to our continuous deployment with 500,000 users that we are operating, and has been up for 18

consecutive months. During our continuous operation, we have observed that during censorship events our system is relied more heavily upon by users. This shows promise for Refraction Networking's use as a more trustworthy fallback technique, to keep users online during censorship events.

1.3 Measuring Web Tracking Evasion

Web tracking is a system in which end-users are often unwitting participants [39, 142, 194, 213]. Third-parties are requested in the background when users visit web pages and these third parties share that information further still, all while building a profile of each user. Users have varied degrees of discomfort with this, depending especially on what sorts of inferences the trackers make, medical and religious information being most sensitive [52] Several browser extensions have been created to help preserve user privacy, and privacy advocates often recommend them [121]. However, researchers lacked quantitative justification for specific recommendations of which tools to install and an understanding of why that tool worked better.

In this study, I first develop a platform for evaluating web tracking across a top-million site list. This is larger than prior work that focused on third-party information sharing, and required a more accurate model of the tools that would block web tracking. I used analysis of blocking extension performance, their performance without Acceptable Ads programs enabled, and qualitative descriptors of the extensions to identify which factors were indicators of the strongest tools.

From these experiments, we now know that organizational mission is the strongest indicator of an extension's efficacy. Users have good reason to mistrust ad blockers' incidental claims to privacy benefits and should focus on privacy-first tools. Developers of these tools also can learn that a focus on high-impact targets will provide more benefit to users at lower cost than expansive block-lists.

1.4 Measuring the Certificate Ecosystem

The last system I measure in this dissertation is the HTTPS certificate ecosystem. Encryption has become more popular as a default means of protecting information transmitted over the Internet, and on the Web that encryption is rooted in certificates. More specifically, a website proves its

identity to users by providing cryptographic proof, a certificate, that its identity a trusted third party verifies. One common way this system fails is the misissuance of these certificates by the trusted third parties: someone other than the operator of `example.com` obtaining a certificate for `example.com` [11]. Such a failure undermines the security of a website when the website operator does not know such a thing has happened.

In this study, I analyze the HTTPS certificate ecosystem from a variety of perspectives, providing the most complete view of all certificates to date. This involved developing new scanning methodologies, i.e. zonefile scanning and CT domain scanning, and comparing passive measurements from the ICSI Notary [13] to a diverse set of scanning methodologies for the first time. We also merged all results from our techniques into ongoing public data sets.

This has helped the average website administrator more easily and completely monitor for misissued certificates, as well as protect that same administrator from errantly issuing certificates. Beyond helping just system administrators, this also helped researchers in Internet measurement better understand the biases in their methodology to help them better serve system administrators. This helps improve the Web's roots of trust.

Thesis Statement. By focusing on end-user concerns in widely deployed systems critical to end-user security and privacy, Internet measurement enables improvements to transparency, trust, and privacy.

Structure. In this dissertation, I demonstrated this by exploring Internet censorship, Web tracker-blocking, and the certificate ecosystem. In Chapters II & III, I explore the application of Internet measurement to Internet censorship, both in providing transparency to a deliberately opaque practice and in aiding censorship circumvention. In Chapter IV, I apply Internet measurement to the evaluation of Web tracker-blocking to produce easily digestible recommendations for laypeople and advice for privacy-enhancing tool designers. In Chapter V, I use multiple perspectives of Internet measurement to illuminate the extent of transparency in HTTPS certificates, improving transparency for the average website administrator, providing insight for improving Internet measurement. Finally, in Chapter VI I draw conclusions from these case studies and look forward to potential uses of

Internet measurement for the direct benefit of end users.

CHAPTER II

Measurement of Application-Layer Censorship

Censorship, as practiced today, is at odds with transparency. A censor's primary goal is to restrict information entering the network under their control and reaching their end users. Furthermore, this is not the only information censors restrict. Censors also restrict the targets of censorship, leading to self-censorship and restricted debate over the practice. The goal of free and open communication on the Internet is held back by a lack of insight into the practices of censors.

Our work's goal is to provide additional insight into censorship practices in a robust, safe, and scalable way. In particular, we look to develop a novel remote measurement technique that detects the last unmeasured censorship technique: application-layer censorship. This technique makes use of existing infrastructure, echo servers, removing the need for user recruitment and improving reliability. We show significant improvements in scale over prior art and validate heuristics that allow for assessment of censorship at a state-level.

This technique has been integrated into the Censored Planet project [35]. As a result of this inclusion, our technique is continuously being used to monitor the deployment of censorship. This proved useful when application-layer techniques were deployed in Kazakhstan in July 2019 [60]. Censored Planet's data, particularly the data derived from our technique, provided information about the composition of block-lists deployed by Kazakhstan. Information like this helped the public debate and led to major browsers taking steps to mitigate their censorship technique [209].

This chapter is adapted from a joint publication that first appeared at the Proceedings of the 27th

USENIX Security Symposium (Sec'18) [200].

2.1 Introduction

Governments often keep specific targets of censorship secret, in order to avoid public accountability or to increase fear and uncertainty [90]. We must measure censorship to gain insights into the deployment of network interference technologies, policy changes in censoring nations, and the targets of interference. Making opaque censorship more transparent illuminates this emerging practice.

Implementing global censorship measurement continues to be a challenging problem. Today, the most common way to characterize censorship uses in-country volunteers to host network probes, such as OONI [75], or to provide anecdotes about what seems to be blocked to monitoring organizations. Both are challenging to scale while providing frequent insight into all vantage points. Moreover, both rely on human volunteers. For individuals living under repressive or secretive government controls, cooperating with security researchers has substantial risks.

An emerging body of work addresses these problems by using existing protocols and infrastructure to remotely measure network interference. Such approaches have been effective in measuring DNS poisoning [162, 180] and for detecting interference in TCP/IP-connectivity between remote machines [69, 161]. There has not yet been a global, remote method for detecting another broadly deployed censorship technique: *application-layer* censorship.

Application-layer censorship has become increasingly important with the rise of content delivery networks (CDNs). CDNs use a small number of network entry-points for a large number of customers, resulting in sizable collateral damage to IP-based blocking techniques. When an adversary wishes to block some, but not all, of these sites, they must look into the content of requests and understand the HTTP or HTTPS headers to determine which site is being requested. This form of blocking is prevalent and effective, but it is not captured by measurements of either DNS or IP connectivity.

In this chapter, we introduce Quack, the first remote censorship measurement technique that

efficiently detects application-layer blocking. Like other remote measurement approaches, we make use of existing internet infrastructure. We rely on servers running protocols that allow the client to send and reflect arbitrary data. This behavior is present in several common protocols, such as in the TLS Heartbeat Extension [181], Telnet servers supporting the "echo" option [165], FTP servers allowing anonymous read and write [183], and the Echo protocol [164]. After identifying compatible servers with scanning, we reflect packets that are crafted to trigger DPI policies. We aggregate instances of reliably detected disruption to identify what and where blocking occurs.

The bulk of our measurements use the RFC 862 Echo Protocol [164]. Echo was introduced in the early 1980s as a network testing tool. Servers accept connections on TCP port 7 and send back the data they receive, making the protocol easy to scan for and to validate expected responses. We find that the public IPv4 address space contains over 50,000 distinct echo servers, providing measurement vantage points in 196 countries and territories. We design and evaluate an echo-based measurement system to test over 500 domain-server pairs per second. The echo protocol also allows us to understand the importance of directionality, cases where blocking is only triggered by messages leaving a region.

The efficiency of our technique allows us to measure application-layer blocking in new detail. We first test 1,000 sensitive domains from our 50,000 vantage points around the world—taking just 28 hours. We find anomalously elevated rates of interference in 11 countries. Each of these countries is reported as restricting web freedoms by Freedom House [78]. We then consider a larger set of keywords in the 40 countries with more than 100 vantage points. We test 100,000 domains, a significantly larger corpus than can be efficiently enumerated by previous techniques. From these experiments, we observe elevated rates of interference for specific domains in 7 countries. These experiments demonstrate the effectiveness of this technique for gaining a fine-grained view of application-level blocking policy across time, space, and content.

Application-layer blocking and deep packet inspection is meant to limit access to targeted content. However, our measurements show evidence of implementation bugs introducing collateral damage. For instance, a health and wellness website is blocked in Iran because it shares part of its

name with a circumvention tool. Other websites with similar content remain available.

By dynamically and continuously test application-layer blocking at global scale, Quack can reveal both deliberately and incidentally blocked websites that have not previously been enumerated. The source code is available online at https://censoredplanet.org/projects/quack.html.

2.2 Related Work

The phenomenon of network censorship first gained notoriety in 2002, when Zittrain et al. [222] investigated keyword-based filtering in China. This initial investigation focused on understanding policy, based off of a single snapshot of content blocking by a single entity.

Both detection and circumvention of censorship remain active problems. Many studies are based on in-country vantage points such as volunteer machines or VPNs, or are one-time and country-specific measurement projects such as studies on Thailand [82], China [41], Iran [15], or Syria [38]. These direct measurements have shown how different countries use different censorship mechanisms such as the injection of fake DNS replies [14], the blocking of TCP/IP connections [208], and HTTP-level blocking [46, 113]. Our measurements are also one-time; however our technique considerably reduces the cost of longitudinal measurement of censorship.

2.2.1 Application-layer Blocking

Many measurement systems measure lists of keywords to test for censorship. In the context of the web, domain names are commonly used as a proxy for services, and are typically drawn either from lists of popular global domains [9], or from curated lists of potentially sensitive domains [40]. Our system uses both of these sources to maximize our comparability, and to test over a sufficiently large corpus.

Detection of keywords more broadly has made use of corpora extracted from observed content deletion, along with NLP and active probing to refine accuracy [45, 81, 220]. Previous systems determining such keywords have largely focused on individual countries and services, especially related to Chinese social media such as Weibo and TOM-Skype [44, 120, 132].

2.2.2 Direct Measurement Systems

Since censorship policies change over time, researchers have focused on developing platforms to run continuous censorship measurements. One notable platform is Tor project's Open Observatory of Network Interference (OONI) [189], which performs an ongoing set of censorship measurement tests from the vantage points of volunteer participants [75]. By running direct measurements, OONI tests are harder for an adversarial network to specifically target. However, these platforms cannot easily certify that it was not the adversary themselves that contributed measurements in an effort to confound results. Moreover, OONI has a smaller number of vantage points, compared to our technique.

2.2.3 Remote Measurement Systems

Academic measurement projects have recently renewed their focus on remote measurement of DNS poisoning [162, 180] and TCP/IP connectivity disruptions [161]. Our system extends this broad strategy to detect application-layer disruption. Our approach provides a uniquely detailed view of the trigger and implementation of interference. We can answer which direction of which packet or keyword was the trigger, and whether interference is implemented through packet injection or dropping. This level of detail is not possible in existing DNS or IP-level side channels.

2.2.4 Investigations of DPI Policies

Deep packet inspection (DPI) and application-level disruption have become standard practice online [51]. Asghari et al. [16] find support for their hypothesis that nations pursing censorship are likely to push deployment of DPI technology. OONI reports on DPI-based censorship in 12 countries with identified vendors, and the Tor project has faced DPI-based blocking in at least 7 countries [6].

2.3 Our Measurement System

2.3.1 Design Goals

Quack is designed to track the use and behavior of deep packet inspection. We focus on four goals:

Detection: Since the specific triggers and behavior of DPI systems are varied and opaque, Quack focuses on detecting when keywords are blocked and the what technical methods are employed. It does not focus on uncovering application-specific grammars.

Safety: Quack is designed to run from a single vantage point, with a goal of worldwide coverage without the need to engage end users to help measure their networks. Instead, our design focuses on the use of existing network infrastructure, in this case echo servers, where the existing protocol reflects network interference information while minimizing risk to end-users.

Robustness: Our system must distinguish unrelated network activity such as sporadic packet loss or other systematic errors that only become apparent at scale from network interference. This goal is achieved by retrying upon indication of failed tests.

Scalability: We aim to accurately measure the phenomenon of keyword blocking on a global scale with minimal cost. This objective is achieved by daily scans for active echo servers, which provide us with coverage of an average of 3,716 autonomous systems daily.

2.3.2 System Design

The Echo Protocol We chose to focus initial measurements on the Echo Protocol. The Echo Protocol, as defined in RFC862 in 1983 by J. Postel, is a network debugging service, predating ICMP Ping. The RFC states, in its entirety:

> A very useful debugging and measurement tool is an echo service. An echo service simply sends back to the originating source any data it receives.
>
> TCP Based Echo Service: One echo service is defined as a connection based application on TCP. A server listens for TCP connections on TCP port 7. Once a connection is established any

data received is sent back. This continues until the calling user terminates the connection.

UDP Based Echo Service: Another echo service is defined as a datagram based application on UDP. A server listens for UDP datagrams on UDP port 7. When a datagram is received, the data from it is sent back in an answering datagram.

There are many active echo servers around the world, including countries known to use DPI. Our vantage points are detailed in Section 2.4.1.

We use echo servers for their defined purpose: measuring transport reliability. We gain additional information about the nature of any unreliability by varying the transport-layer data and observing differences in the network's behavior. This affords us insight into the nuanced network perspectives of remote hosts, contributing to the exposure of national censorship policies.

We take advantage of three features of echo that lend themselves to our purposes. First, the protocol has a well defined response to every request, which makes the classification of abnormal responses trivial. Second, due to the ability to send arbitrary binary data, we can test censorship of any application-layer protocol that utilizes TCP or UDP as its transport protocol. In this chapter, we focus on HTTP and HTTPS. Finally, because echo servers reflect content back to our measurement machine, we are able to also detect censorship in the outbound direction, and differentiate it from censorship triggered by our inbound request. Direction-sensitive interference is a known capability of modern DPI boxes. Figure 2.1 illustrates the Echo Protocol in the absence of noise.

If, unlike in Figure 2.1, the middlebox injects a non-RST response to the echo server, we are still able to observe the interference. In fact, we are able to see the injected message because the echo server will echo the content it observes back to our measurement machine.

We note that echo is not the only protocol that can be used for this technique. We focus on it here because it provides a clear signal, but more scale can be achieved by extending measurements to any other protocol where an expected response will occur when client probes are sent.

Defining A Trial We call an individual transaction with a remote server a trial. A trial is conducted with a single server, using a single keyword, and with a single application protocol containing that keyword. For example, consider `example.com` as a keyword wrapped within the format of an

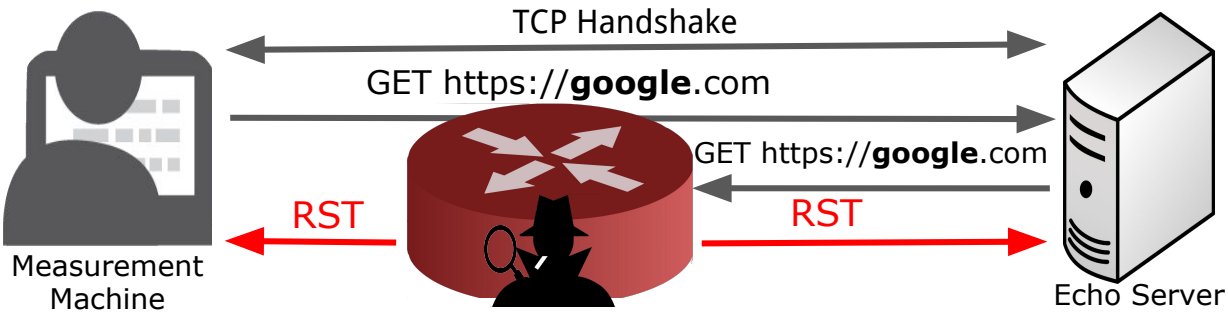


Figure 2.1: **Echo Protocol**—The Echo Protocol, when properly performed, is a simple exchange between the client and server where the server's response is identical to the client's request. In the example above, the censoring middlebox ignores the client's inbound request, but reacts to the the echo server's response, injecting RST packets and terminating the TCP connection.

HTTP/1.1 request.

During a trial, we initialize a connection to the server and send it the formatted keyword. We read the response, and pause for a short period. Finally, we send a short, innocuous payload to verify that the connection remains active. If the server responds the connection is closed successfully, we consider the trial a success.

The pause is necessary to allow injected RSTs by interference technology to reach either host in the connection. This gives us the ability to directly identify that an interfering network is attempting to exploit a race condition via a Man-on-the-Side deployment. By verifying that the connection is still open after the keyword is sent, we ensure that there is not asymmetric interference occurring, in which the interfering network closes the connection or begins dropping packets to our measurement machine.

Test Phases The Echo Protocol enables trivial disambiguation between correct and incorrect responses, but distinguishing noise from network interference requires additional effort. The Internet is by definition best-effort, and therefore even in the face of no interference, there will be failed connections with echo servers. Additionally, interference technologies are themselves imperfect, meaning that some trials will be successful even when the data is typically disallowed, for example when the DPI boxes are overloaded [70].

Quack is designed to extract meaningful signal from the noisiness of the network. We think about this as validating signs of failure through additional measurements, but there is a trade-off:

Not retrying would lead to many false positives, resulting in an inflated rate of interference. On the other hand, many retries increase false negatives as sensitive connections slip past interference technology and are categorized as successful. We choose to be conservative in our designation of interference, designing our system to minimize false positives by retrying failures several times.

Our implementation designates a "test" as the repeated trial of a particular server and keyword. A test proceeds in three phases, as shown in Figure 2.2:

Retry Phase First, we run a trial with the keyword and retry if it fails. We end the test as soon as we have a successful trial, and declare the test a success. We expect interference to be sparse. For example, the highest failure rate we observed when testing sensitive keywords in a country known to implement interference was 2.2% of tests not ending in success after the first trial. We allowed up to 5 retries in our experiments.

Control Phase After five trials have failed, we progress to the Control Phase. In the Control Phase, we trial an innocuous keyword. If the server successfully completes this trial, we conclude that the five previous failures were due to network interference. If the control keyword fails, we proceed to the final phase. In our experiments, we use `example.com` as our control keyword.

Delayed Phase Finally, we account for stateful disruption. This is observed, for example, in China [212]. We test for this behavior by performing another innocuous trial after a delay. If this trial succeeds, we classify the keyword as sensitive. If it fails, we mark the test as No Result. This may occur if the echo server becomes unresponsive during our test.

We use a two minute delay determined empirically. Knowing that some middleboxes perform stateful blocking, we test every server in censoring countries with an HTTP request for the most commonly censored domain in that country. Then, we attempt to reconnect every 10 seconds with an innocuous payload until we succeed. The resulting distribution in Figure 2.3 shows 120 seconds is a sufficient delay.

These steps ensure that Quack is robust and can distinguish unrelated network activity, such as sporadic packet loss and other systematic errors, from deliberate forms of network interference.

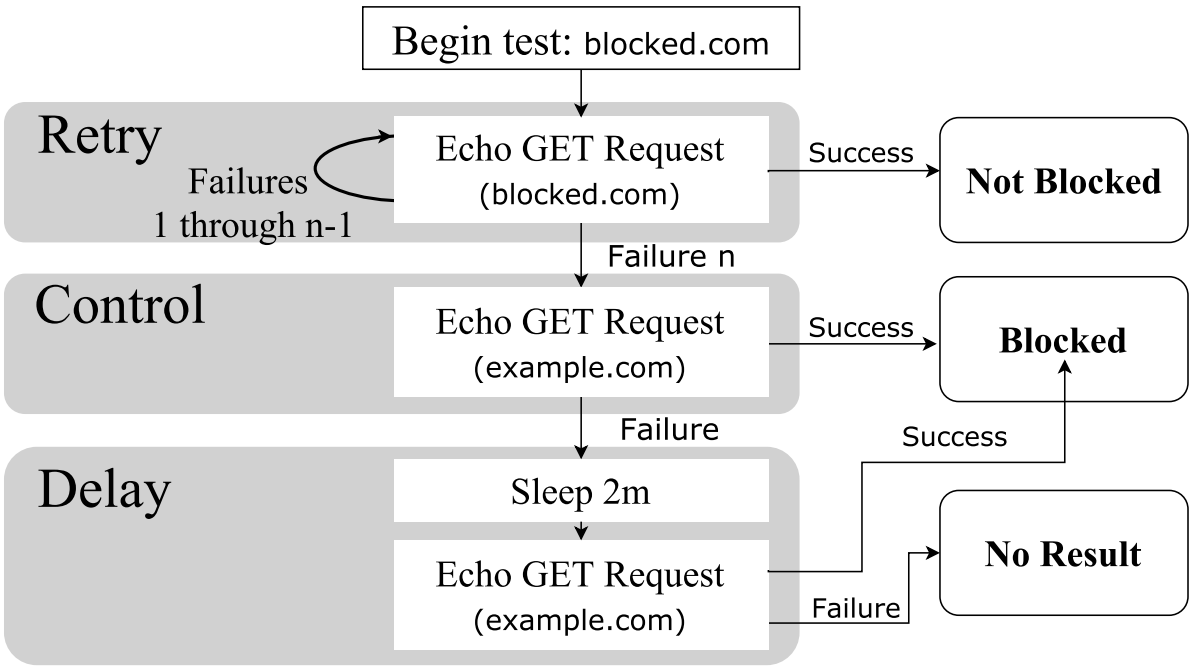


Figure 2.2: **Test Control Flow**—A single test using an echo server is performed by following this diagram. The most common path is also the fastest, in which an echo server responds correctly to the first request and the test is marked as Not Blocked. If the server never responds correctly, the experiment is considered a failure and we do not use the test in our evaluation.

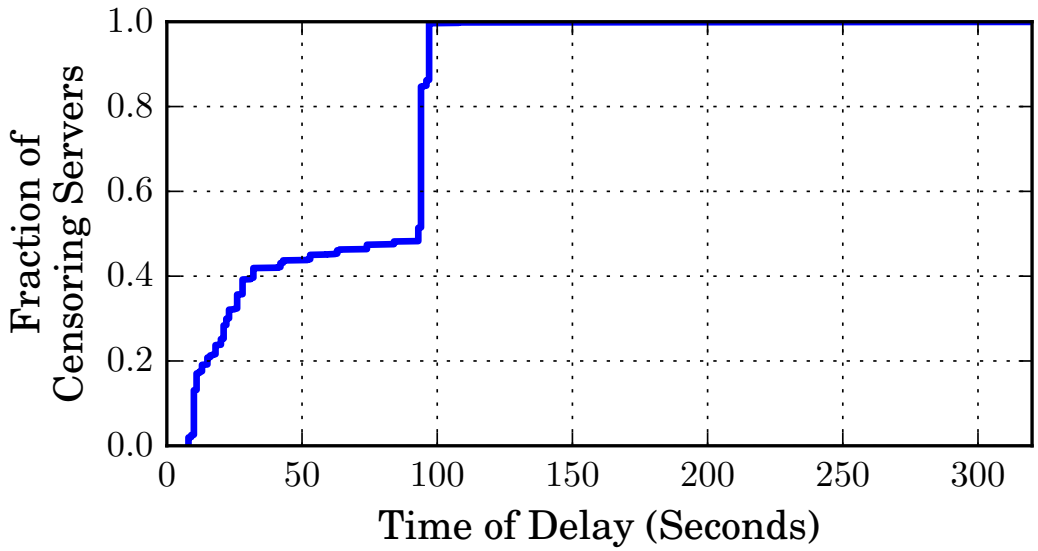


Figure 2.3: **Persistent Interference Duration**—We use echo servers in all countries we observe censorship to empirically measure the length of time interference occurs after a censorship event has been triggered. Roughly half of the servers responded correctly to our request within 60 seconds. By 100 seconds, 99.9% responded correctly. We therefore choose two minutes as a safe delay in the Delay Phase.

Classifying Interference Although we conduct multiple trials within a test, false positive tests can still occur. We do not categorize a single failed test as interference, since it could be due to temporary routing issues or other transient failure. Even if the test is representative of policy, we wish to differentiate interference that is occurring at a local level, such as a corporate firewall, from that implemented at a national or regional level. To address both of these, we consider all tests in a country, comparing keywords by the rate of tests yielding a Blocked result. This allows us to observe the phenomenon of blocking at a country level.

This last layer of aggregation is formed by calculating a "blocking rate" for each keyword-country pair, equal to the number of tests classified as Blocked divided by the number classified as either Blocked or Not Blocked. Effectively, this removes No Results from our analysis. Prior work that has looked at failure rates aggregated across servers has required a minimum number of trials in an aggregated group to report on the blocking rate for that group [180]. We follow this convention, as it is consistent with our design goal of *Robustness*. Selecting a threshold for the number of experiments that is too low reduces our confidence, while selecting a threshold that is too high excludes more countries. Upon manual inspection of the number of servers in countries reported to perform blocking, we determine 15 as threshold that balances *Robustness* and the inclusion of anecdotally blocking countries. In Section 2.5.1, we validate the countries in which we observe widespread censorship using external evidence.

Due to No Result tests and echo servers churning out of our test set, the keyword blocking rates in a given country have many possible values. To approximate the probability density function of the keyword blocking rates in a country, we count the number of blocking rates in n even intervals over $[0, 1]$, where n is configurable. Having this approximated distribution in each country of keyword blocking rates lets us consider each keyword's failures in the context of the country's noise. We can also categorize each country based on its distribution.

When there is no blocking, we assume Blocking events due to noise are independent and only occur with very small probability. We confirm this in Section 2.5.1. Since the probability of failure due to noise is so small, given our redundancy in each test, we would expect that our approximated

distribution of the blocking rates be monotonic in the case that there is no blocking. In our control experiments with no expected interference in Section 2.5.1, we find all distributions to be monotonic, and we empirically find the blocking rate to be 0.01%.

We mark interference in countries whose distribution of keyword blocking rate is not monotonic. More precisely, we say that the keywords whose blocking rates are in the interval that breaks the monotonic trend and those keywords with higher blocking rates experience interference in that country.

We considered several trade-offs when choosing the number of intervals, n. We do not want an n larger than the minimum number of tests per keyword, 15, because this could cause consecutive numbers of blocking results to be in the same interval, creating an artifact in the distribution. However, we want as many buckets as possible, so that our smoothing does not remove too much of the detail of the distribution. To balance these concerns, we use $n = 15$ buckets consistently for the rest of our analysis.

We implement a system in Go 1.6, utilizing light-weight threads for parallelism. We restrict ourselves to one concurrent request per echo server, to restrict load on the echo server, and at most 2000 total concurrent requests. Our test server was able to process 550 requests per second and has a quad-core Intel E3–1230 v5 CPU, 16 GB of RAM, and a gigabit Ethernet uplink.

While we initially ran tests with our measurement machine source port set to 80, in order to appear more similar to real HTTP connections, we found no difference in our results while using an ephemeral source port. Using an ephemeral source port also allowed us to follow standard conventions and to host an abuse website on the standard HTTP port of our measurement machine.

2.4 Experimentation

2.4.1 Echo Server Characterization

In order to better understand any biases inherent in our data, we first characterize the population of echo servers we make use of in our study.

2.4.1.1 Discovery

To discover echo servers in diverse subnets and geographic locations, we perform Internet-wide scans with the ZMap toolchain [59] on the IPv4 address space. We ran daily scans for two months, between June 1st to July 31st, discovering more than 50,000 echo serves each day.

Upon discovering hosts that respond to our SYN packets on port 7, we initiate connections to the potential echo servers. We send a randomly generated string and verify that they reply with an identical string. During our first trial, we find that $57,890$ servers reply with the correct string, over $3,766$ ASNs. Many of our experiments take place over the course of a day, so we measure the coverage of echo servers that reply 24 hours later. We find 92% of ASNs have an echo server that is online during this second test.

In Table 2.1, we show the number of servers still online after 24 hours, which is significant because our experiments run over the course of a day. Only those servers that are stable for at least 24 hours will test all keywords in the experiment. We observe that this reduces the diversity of our coverage, but not significantly, and note that this biases our results towards stable echo servers.

2.4.1.2 Churn

We looked at our daily scans in order to understand how stable echo server IP addresses are over time. While an average of 17% of echo servers churn away from their IP address within 24 hours, we observed that 18% were stable and responsive throughout the entire duration of our measurement. Additionally, the rate at which echo servers churn decelerates, so the first day reports the largest

Server Set	IP Addresses	/24s	ASNs	Countries
SYNACK	5,260,118	109,729	6,932	198
Echo	57,890	38,977	3,766	172
Stable (24 hr)	47,276	31,802	3,463	167

Table 2.1: **Discovery of Echo Servers**—Server discovery is a staged process. A ZMap scan discovers servers that SYNACK on port 7, but we find that most of these servers will fail to ACK or will RST when receiving any data. To remove these misbehaving echo servers, we attempt to send and receive a random string to all SYNACK servers, giving us the set of functioning echo servers. Of these, 47,276 remained Stable over 24 hours, making them useful for long running experiments.

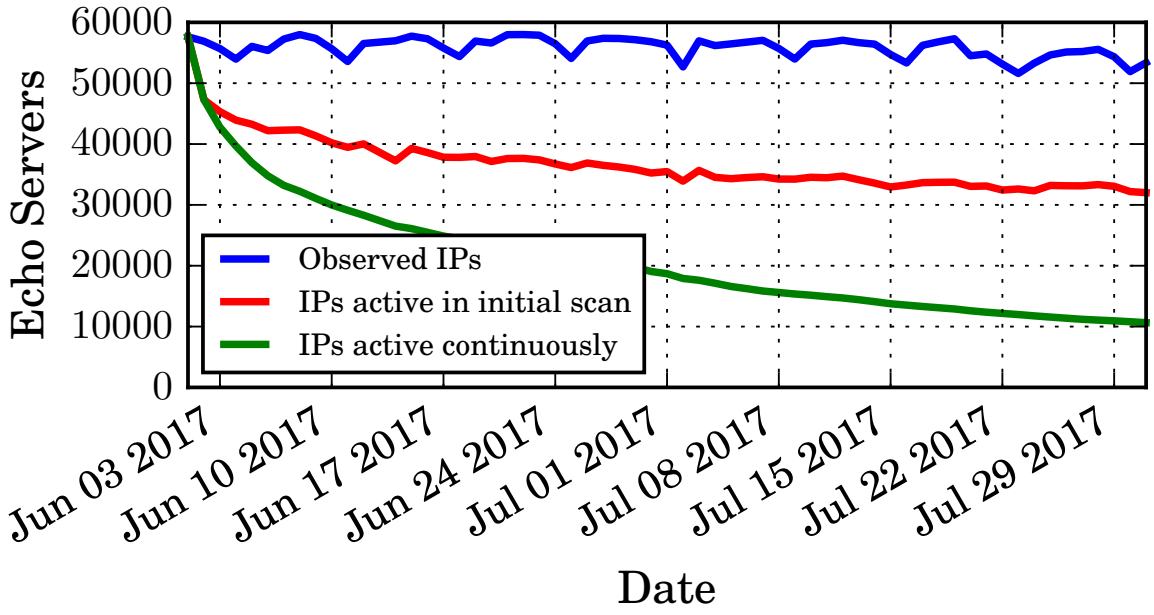


Figure 2.4: **Echo Server Churn**—Only 18% of tested servers were reachable in every observation over 2 months of daily scans. However, 56% were present in both our first and final scans.

churn rate across the study.

Echo servers not only churn out of the set of IP addresses from a given day—they also churn back in, as shown in Figure 2.4. While we only observed 18% of echo servers from our first discovery scan in every daily scan, 56% of echo servers from our first discovery scan were also in our final scan 61 days later.

2.4.1.3 Identification

To understand the composition of machines running echo servers, we randomly selected 1% of responding echo servers on July 17, 2017. For this sample, we performed OS detection on each IP address using Nmap. The most common system families as defined by Nmap are shown in Table 2.2. There were 56,228 working echo servers on this date. Of the 562 we tested, Nmap identified 463 (82.4%) of the operating systems. Nmap reported a median accuracy of 99% for the identifications. This test covered 54 countries.

Of the echo servers we scanned with Nmap, 251 (44.7%) had full device labels containing the words “server”, “router”, or “switch”. Of the remaining echo servers, 70 (12.5%) were Linux,

and 26 (4.6%) were Windows. The rest were identified as various other systems such as firewalls, controllers, and embedded systems. In total, 4% of echo servers were given device labels that left doubt as to whether they were infrastructure machines, because they were identified as non-server Windows machines, and 2 devices were identified as running Android. It would be infeasible to run Nmap's OS detection service against all echo machines, but we do not believe that to be necessary to safely use all functioning echo servers, as we discuss in Section 2.6.1.

2.4.1.4 Coverage

Echo servers provide us diverse vantage points in a majority of countries. We associate IPs with autonomous systems using the publicly available Route Views dataset [175], and locate each server to a country using the MaxMind GeoIP2 service [138].

On average, we observed echo servers in 177 countries. Of these countries, we observe an average 39 countries with more than one hundred echo servers and 82 countries with more than fifteen echo servers. This provides insight into a large number of countries.

We compare our method's coverage with that of the OONI project [75], which enlists volunteers worldwide to run scans from local devices to measure network disruption. OONI makes this data public with the consent of the volunteers, but probes do not have unique identifiers; therefore, we use the of number of distinct autonomous systems per country to estimate coverage.

We compared the number of unique ASes observed for both tests during the week of July 8–15, 2017. As shown in Figure 2.5, echo servers have a much more diverse set of vantage points and

OS Family	**Echo Servers**
Windows	180 (32.0%)
Embedded	139 (24.7%)
Linux	71 (12.6%)
Cisco IOS	38 (6.8%)
Unsuccessful identification	99 (17.6%)
Other	35 (6.2%)

Table 2.2: **Identification of Echo Servers**—We scanned 562 (1%) echo servers with Nmap's operating system detector on July 17, 2017 and found that the most of the echo servers were either Windows machines or embedded devices, as identified by Nmap. This scan yielded a median accuracy of 99%.

over a larger number of countries. During the week of our comparison, OONI data was available for 113 countries, while echo servers were responsive in 184. Furthermore, the total number of ASes seen in the echo measurements was nearly an order of magnitude larger than that of OONI: we observed echo servers in 4458 unique ASes; OONI measures 678. While OONI probes provide rich measurement for the locations they have access to, our technique providers broader and more consistent measurements.

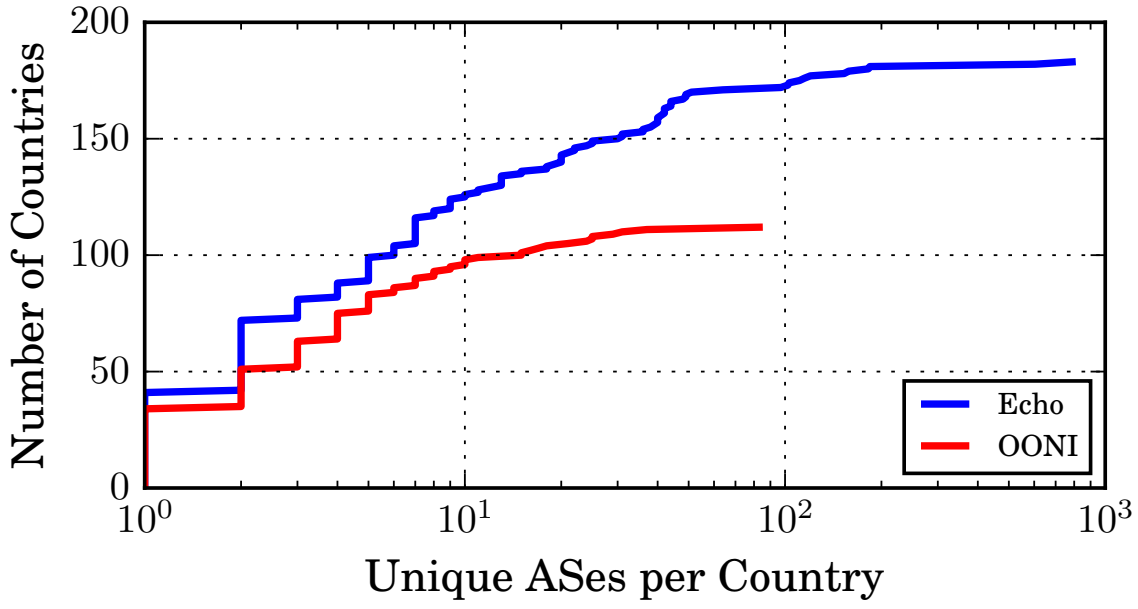


Figure 2.5: **Coverage of Autonomous Systems per Country**—Echo servers were present in 184 countries with 4458 unique ASes, while OONI probes were in 113 countries with 678 ASes.

2.4.2 Datasets

In our study, we examine URLs as the source of content that may be disrupted. In our experiments, unless specified otherwise, we send the domain name in the context of a valid HTTP/1.1 GET request. This allows us to observe a particular subset of application-layer interference, and one that is well documented [45].

Control We first perform a control study. To do so, we test a number of innocuous domains as our keywords, which are expected not to be censored, and repeat them against every echo server. The domains we choose are of the form `testN.example.com` with incrementing values of N. We

perform this experiment 1109 times per server. Since there should be no artificially induced network interference, we can validate our technique using the results of this study. This test was performed July 20–21, 2017 from our measurement machine inside of an academic network.

Citizen Lab We use the the global Citizen Lab Block List (CLBL) [40] from July 1, 2017 as a list of keywords to run against all echo servers. This list has 1109 entries. It is curated by Citizen Lab to provide a set of URLs for researchers to use when they are conducting censorship research. Significant difference between this test and the previous test indicates that our system is capable of detecting application-layer interference of the domains in this list. This test ran on July 21–22, 2017, from our measurement machine.

Discard We then repeat the Citizen Lab study using a closely related protocol, the Discard Protocol [163]. The Discard Protocol is designed similarly to the Echo Protocol, but instead of echoing back any received data, it is simply discarded. By repeating our experiment with discard, we can determine if existing middleboxes detect keywords that are seen inbound to its network. If this were the case we would see the same interference in the Discard Protocol as the Echo Protocol. Otherwise, we will be able to determine that interference technologies do notice the direction of sensitive content. This test run on July 19–20, 2017, from our measurement machine.

TLS This study demonstrates the application-layer flexibility of our technique. We perform the Citizen Lab experiment again, but instead of embedding the Citizen Lab domain list in valid HTTP request, we place the domain in the SNI extension of a valid TLS `ClientHello` message. This will allow us to discern what difference exists between interference of HTTP and HTTPS. This test ran on July 23–24, 2017, from our measurement machine.

Alexa Top 100k Finally, we use our system to test the top 100,000 domains from Alexa [9] downloaded on July 12, 2017. This is a set of domains orders of magnitude larger than that of prior works studying application-layer censorship. To achieve full measurement of such a large set of domains, for each domain we select 20 servers in each country. Additionally, we restrict our test to the 40 countries with more than 100 echo servers. This test demonstrates most of all that our tool can be used at scale for significant research into application-layer blocking at a country granularity.

This test ran on July 25–28, 2017, from our measurement machine.

2.5 Evaluation

In this section, we provide the results of the studies described in Section 2.4. Our evaluation provides support for the Quack's practicality as an application-layer measurement tool in two ways. First, we describe what behavior our measurements detected given a set of URLs known to be censored, in order to verify that our results correlate with previously observed phenomena. Then, we support our claim that our system works at scale, and present the results of an experiment that measured a larger corpus of domains across a greater number of countries than any previous study.

2.5.1 Validation

We control for noise, non-protocol-compliant servers, and other anomalous behaviors by measuring echo server behavior using innocuous domains of the form `test`*N*`.example.com`. Mock queries to these domains are used to demonstrate behavior in the absence of disruption, since these domains are unlikely to be blocked. This allows us to identify a baseline for ordinary network and echo server failure when interacting with each remote network, and understand our subsequent test results in light of a baseline model of expected behavior.

The first assumption we make in designing our control tests is that the class of domains `test`*N*`.example.com` will face no blocking by the network between our server and the echo server. To validate this assumption, we perform a set of measurements to all echo servers using only this control class of domains, and consider the failure percentages we observed. We show the distribution of failures per domain tested in Figure 2.6.

We observe a median domain failure rate of less than 0.01%, and a maximum failure rate across 1109 domains of 0.08%. Additionally, the domains in the upper quartile of disruption rates are evenly distributed over the class of innocuous domains, independent of the value of *N*.

Using the technique described in Section 2.3.2, we classify no country as interfering with any of our control domains. We also confirmed these results using another control domain: `echotest.umich.edu`, validating our control.

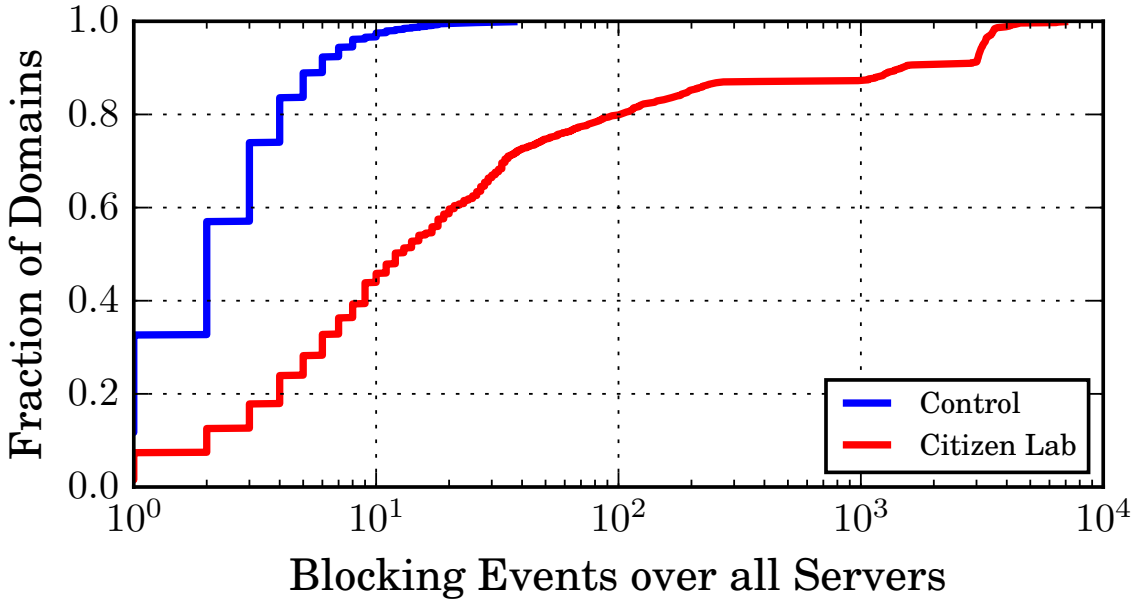


Figure 2.6: **Keyword Reliability**—Each of 1109 domains were sent to 54,515 echo servers for the Control and 54,802 for the Citizen Lab experiment. We count the blocking events per keyword, observing that the largest blocking rate for a given keyword was 8.5% in CLBL and 0.08% in the Control. This supports our hypothesis that these domains are sensitive.

We assume failures in the absence of network interference are independent of which server is used. This allows us to present a distribution for the null hypothesis that is independent of either variable, and therefore constant. A few factors could cause a given server to fail many innocuous domains: network unreliability, echo server unreliability, or actual blocking occurring for our innocuous domains. Despite this, in Figure 2.7 we see that 98% of servers see no blocking events.

We observe that during the duration of our experiment, 17% of echo servers appear to churn away, which is indicated by their yielding two No Result tests sequentially. This is roughly as many as we observe churning away in a day for our discovery scans. This confirms that our results will be biased toward networks with stable echo servers.

Finally, we empirically determine how long measurements should wait when a blocking event is detected in order to allow stateful DPI disruption to disengage. Shorter timeouts will allow us to test more domains against a given server in a shorter time, while longer timeouts are less likely to incorrectly classify a domain as a failure due to a previous sensitive domain having triggered

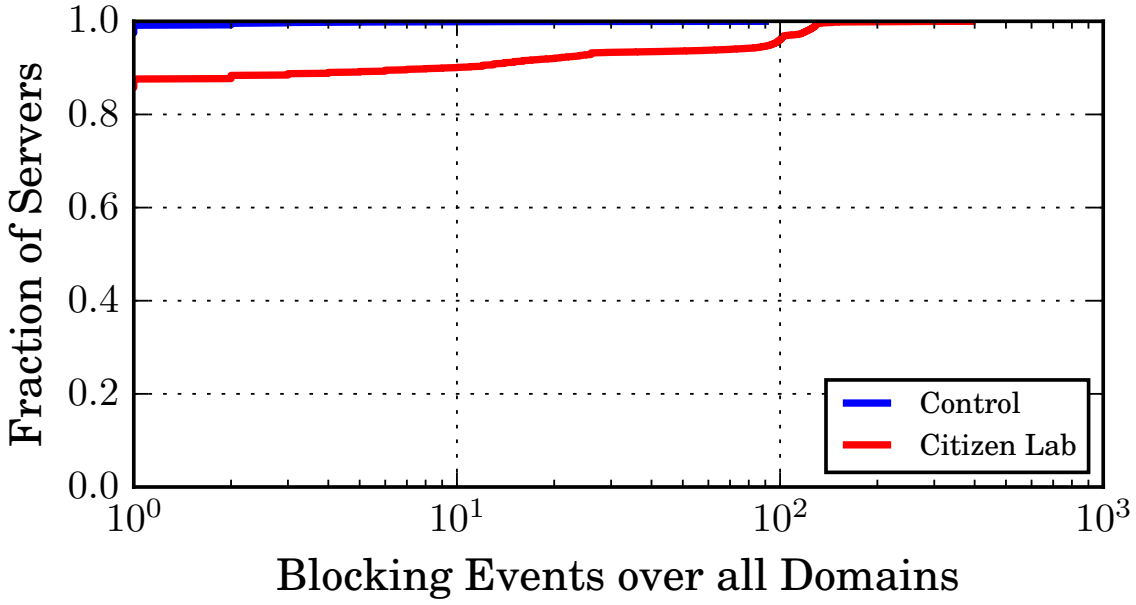


Figure 2.7: **Server Reliability**—For both the Control and Citizen Lab experiments, we send 1109 mock HTTP requests to all echo servers. We find that 98% of servers never resulted in a blocking event in the Control experiment. We observe significantly more blocking among a small set of servers in the CLBL test. This demonstrates that interference occurs with very few hosts.

stateful blocking. Our system as implemented is not fundamentally limited by a longer timeout, because there are more servers to test at any given time than there are servers waiting for that timeout to expire. As such, the two-minute delay we empirically determined as shown in Figure 2.3 is a minimum, and the system may take longer to schedule the subsequent trial in a test against a disrupted server. We observe that all delays were less than five minutes in practice.

2.5.2 Detection of Disruption

Next we test each of the domains on the Citizen Lab global list against all echo servers by formatting them as valid HTTP GET requests. We expect to see interruption in this test because the Citizen Lab domains are known to be blocked in countries around the world. This is confirmed by the difference to the control in Figure 2.6 and Figure 2.7.

Using our method of classifying interference as described in Section 2.3.2, only 12 countries of 74 tested against all domains demonstrate evidence of keyword blocking in this test. The interfering countries, number of domains for which we observe interference, and what categories those domains

are contained in are given in Table 2.3.

For each country we list in Table 2.3, we look for external evidence to support the conclusion that we observe government-sanctioned censorship. One source of external evidence is the Freedom on the Net report by Freedom House [78]. Of the countries in the table, nine are rated as "Not Free" and two are rated as "Partially Free."

South Korea and Jordan are those listed as Partially Free by Freedom House; however, both are indicated in ONI's most recent country profiles as performing filtering [156, 157]. In the case of South Korea, blocking based on HTTP request content is specifically identified. In further support of the observed phenomenon being action at a national level, the echo responses in South Korea that did not match the echo requests were HTTP redirects to a government-run website outlining the reason the requested domain was blocked. This is another advantage of the Echo Protocol — we are able to see the responses injected to the echo server, because they are then echoed back to us.

Two countries were identified by our system as having a significant proportion of blocking, but had no evidence from other sources that there would be restrictions on the Internet: Ghana and New Zealand. Ghana is not evaluated by Freedom Net, but the Department of State stated in its 2016 Human Rights report [193] that there were no governmental restrictions to the Internet. Upon inspecting the scope of blocking, in both cases, it is restricted to a single academic network in the country, and all echo servers in that AS reported interference. In all other countries identified by our system as performing blocking, we observe interference in more than one AS. Our technique is not fine-grained enough to detect censorship across all networks, and in these cases we have visibility into only a few locations that have close proximity. For these reasons, we exclude Ghana and New Zealand from Table 2.3.

While this presents a case that the interference we identify is genuine, we do not claim that we identify all genuine interference. The list of all countries with at least 15 echo servers is presented in the Appendix. This list has multiple other countries that are listed as "Not Free" in the Freedom of the Net report, including Belarus, Russia, Pakistan, and Vietnam.

Pakistan, as an example, is identified by prior work [180] as practicing DNS poisoning. DNS

Country	HTTP	Discard	TLS	Top Categories
China	126	126	0	NEWS, ANON
Egypt	6	5	2	ANON, NEWS
Iran	25	0	374	PORN, LGBT
Jordan	8	1	4	ANON, NEWS
Kazakhstan	4	0	0	MMED, FILE
Saudi Arabia	2	0	0	NEWS, ANON
South Korea	14	0	0	PORN, GMB
Thailand	11	0	0	PORN, NEWS
Turkey	12	14	14	ANON, NEWS
UAE	8	0	17	NEWS, COMT
Uzbekistan	1	—	1	MISC
Union	220	146	435	NEWS, ANON

Table 2.3: **Interference of CLBL**—We perform multiple experiments to measure interference of domains in the Citizen Lab global block list. Quack detected keyword blocking in 13 countries, with 220 unique domains blocked in our simple HTTP experiment. There is little intersection between different countries, and only 20% of tested domains exhibited interference anywhere. Category abbreviations are defined in the Appendix.

poisoning is one potential implementation of Internet censorship, and would render application-layer blocking unnecessary. The technique presented in this chapter does not consider any other possible implementations of Internet censorship, and will therefore miss countries who do not rely heavily on application-layer censorship. Furthermore, many non-technical factors are included in the Freedom of the Net rating; not all "Not Free" countries block content using technical means.

We have validated our classifications with anecdotal reports, but we also want to ensure there is consistency in our classification. To do so, we look at what percent of ASes, /24s, and echo servers in a given country observe any Blocked result in this experiment. The countries that we observe widespread blocking in are represented in the shaded region in Figure 2.8. While some countries have interference in almost all instances, e.g. China, there are several countries with interference not performed across the entire country. This potentially reflects heterogeneous deployments of interference. We observe in Figure 2.8 that some countries that we do not classify as blocking any domains have comparable numbers of servers experiencing at least one Blocked result as countries we do classify as blocking. These countries, Mexico and Zambia, have blocking events that are disperse and inconsistent in the set of domains being blocked, reflecting either unreliable echo servers or echo servers with highly-local blocking. Additionally, these countries had "no reports of

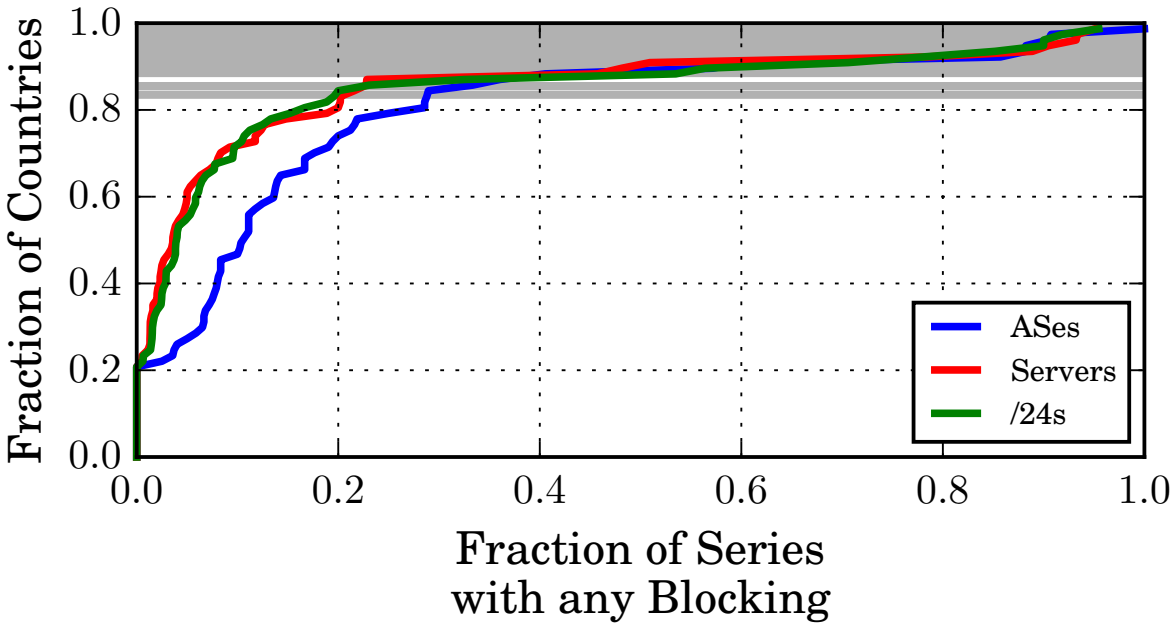


Figure 2.8: **Blocking Rates Per Country**—We examine the CLBL results, looking at what fraction of ASes, Servers, and /24s in each country observe any Blocked result. The shaded regions are countries we identify as having widespread interference. While some countries face near ubiquitous interference across tested servers, more countries display large variation.

blocking" in the Freedom of the Net 2016 report [78].

The most commonly blocked domains we observe in the Citizen Lab block list are shown in Table 2.4. The most commonly blocked domain is `www.hotspotshield.com`, the homepage for a free VPN service. VPNs are common circumvention tools. Surprisingly, it is only blocked in five of the 13 countries where we detected censorship: China, Iran, Jordan, Turkey, and UAE. We see that the most consistently blocked domains are for circumvention tools, pornography, and gambling. Political content tends to be region-specific, and is less often blocked by multiple countries.

2.5.3 Disruption Mechanisms

By using echo servers, we ensure that the potentially sensitive payload is on both the inbound and outbound halves of the connection. This means that our system will detect interference regardless of directionality of the censor. In order to test whether the direction of the request matters, we perform the Citizen Lab test using the Discard Protocol [163]. This protocol is similar to the Echo Protocol,

but instead of echoing the request, the server only ACKs the data. Blocks that occur in our test of echo servers, but not discard servers, could be instances of blocking on only outbound data. This test provides additional valuable insight into the mechanisms used for blocking.

We test the subset of echo servers that are also discard servers, sending identical payloads as in Section 2.5.2. Echo servers are also often discard servers, so this requirement reduced the number of testable servers from 57,309 to 27,966. Of the 11 interfering countries, we are able to maintain enough servers to classify disruption in all but Uzbekistan.

In the 10 remaining countries we observed blocking when using echo servers, we continue to observe disruption in only 4 when using the Discard Protocol: China, Egypt, Jordan, and Turkey. This implies the other countries we observe performing HTTP blocking are doing so only on data outbound from their network. This evidence is not necessarily conclusive, as the reduced set of echo servers may be reducing our visibility into these countries. For example, we observe reliable disruption in a few Iranian ASes for the Discard Protocol. However, because the vast majority of Iranian ASes do not interfere in this test, we do not classify the interference as widespread across the country.

2.5.4 HTTP vs HTTPS

The Echo Protocol allows arbitrary data to be sent to and returned by the echo server. This flexibility is a strength of our technique, and is an advantage over other protocols with more constraints on sending and receiving arbitrary byte streams. To demonstrate why this capability is important, as well as illuminate practices in network interference, we repeat our test of the Citizen Lab Block List, but send requests formatted as valid TLS ClientHello messages with the Server Name Indication (SNI) Extension.

The Server Name Indication Extension [61] was developed to allow a TLS client to inform the server what domain it is attempting to connect to before the server must send a certificate. Since certificates are used for authentication and linked to domain names, a server hosting many websites would need this information to connect to a client securely. Unfortunately, SNI places the domain name in clear-text in the first message sent by the client to the server, making it easy to detect when

Domain	Blocking Countries	Category
www.hotspotshield.com	5	ANON
www.xvideos.com	4	PORN
www.pornhub.com	4	PORN
www.gotgayporn.com	4	PORN
bridges.torproject.org	4	ANON
www.pokerstars.com	3	GMB
adultfriendfinder.com	3	DATE
www.torproject.org	3	ANON
www.wetplace.com	3	PORN
ooni.torproject.org	3	ANON

Table 2.4: **Top Interfered CLBL Domains**—We compared the list of domains interfered with in each country to find those most broadly blocked. The top 10 are presented above. Pornographic websites are overrepresented in the table, but the single most broadly blocked domain is the homepage of a free circumvention technology. China blocks every domain in the table.

a client is connecting to a particular site from only the first message in a TLS handshake. We find that networks do interfere based on this first message alone.

Of the 12 interfering countries we detect in the Citizen Lab experiment, we were able to conduct enough tests to confidently classify all of them. We continue to observe disruption in only 5 when using TLS: Egypt, Iran, Jordan, Turkey and UAE. For the other countries in Table 2.3, TLS may aid in circumventing interference of HTTP requests based on the application-layer.

The only instance of interference occurring in a country that was not detected with just HTTP requests from the Citizen Lab list occurs in New Zealand. The domains blocked are identical across two servers in the same /24 routing prefix, which is allocated to an academic institution in the country. We conclude that the blocking is being performed by the institution, and not a national policy decision to only block HTTPS.

While the domains we observe interference with are similar in four of the five countries, in Iran the set of disrupted domains grows drastically when testing with TLS ClientHellos: the number of blocked domains in Iran increases from 25 to 374. The list of blocked URLs also changes composition to include significantly more domains classified by Citizen Lab as News, Human Rights, and Anonymization tools.

There are several possible reasons a country would implement a policy blocking a domain

through HTTPS but not HTTP. As the domain name is the only visibility into the nature of the content in a HTTPS connection, a country could be aggressive in blocking domains where only a single page on the domain is undesired. In the case of HTTP they could simply block the specific page or given keywords, since all of the content will be visible to the censor. Alternatively, a country could wish to have visibility into the resources accessed at a given site, which forcing a downgrade to HTTP would allow.

2.5.5 Disruption Breadth

We have established to this point that we have a tool that allows us to test for application-layer censorship across 74 countries for roughly a thousand domains. While this is useful, we explore a different capability of our tool in this section. We perform a search for disruption across 40 countries for the 100,000 top domains as ranked by Alexa [9].

In order to perform tests across this many domains, we restrict ourselves to at most 20 requests per domain per country; this reduces the total number of requests dramatically. Several countries contain thousands of echo servers. Additionally, because we only make serial requests to any particular server, we test only in countries with at least 100 servers. This means the most requests a server must process sequentially is 20,000.

This experiment reveals interference in 7 countries, presented in Table 2.5. Of the countries with enough echo servers to be tested, the countries we observe blocking the top domains are the same countries who were blocking domains in the Citizen Lab experiment.

Of the domains that are similar in both the Citizen Lab list and the Alexa Top 100k, we see large overlap in blocked domains. We define similar domains as those with the same domain name, not including sub domains.

One interesting behavior this heuristic shows is in Egypt. Several `torproject.org` subdomains are tested in the CLBL, but only the root domain was tested in Alexa. We observe that the interference in Egypt is dependent on subdomain: the root domain `torproject.org` is not blocked, and the subdomain `www.torproject.org` is blocked on one echo server in Egypt when tested only seconds apart.

Country	Domains Blocked	Unique	Citizen Lab
China	787	712	146
Egypt	27	20	1
Iran	1002	795	10
Saudi Arabia	3	2	1
South Korea	1572	1139	15
Thailand	38	16	0
Turkey	291	120	7
Union	3293	—	180

Table 2.5: **Interference of Alexa 100k**—We test the Alexa Top 100k domains across the 40 countries with the most echo servers and observe censorship in 7. The number of censored domains in the Alexa list does not necessarily correlate with the number blocked in the CLBL, but every country seen blocking in the Citizen Lab experiment also interferes in the Alexa 100k.

Category	Blocked Domains Not in CLBL
Pornography	2085 (99%)
News and Media	114 (92%)
Search Engines and Portals	100 (98%)
Information Technology	85 (97%)
Personal Websites and Blogs	85 (50%)
Proxy Avoidance	59 (87%)
Shopping	36 (100%)
Other Adult Materials	35 (90%)
Entertainment	33 (97%)
Streaming Media and Download	31 (86%)
Uncategorized	89 (96%)
Other	378 (94%)

Table 2.6: **Alexa Domain Discovery**—We categorize the domains blocked in each country in our Alexa 100k experiment, excluding those with a similar entry in the Citizen Lab experiment, and present the top 10 categories. As in other experiments, the largest censored category is pornography. However, other categories show the breadth that can be uncovered by testing the entire Alexa 100k. For example, none of the blocked shopping domains in the Alexa dataset were in the CLBL.

Another interesting blocking behavior we observe is that Iran blocks an innocuous health and lifestyle site, `psiphonhealthyliving.com`. This site is likely collateral damage, as Iran also blocks the domain `psiphon.ca`, the homepage for a censorship circumvention technology. Additionally, we can observe that in Iran, all domains belonging to the Israeli TLD (`.il`) are blocked.

Testing the Alexa 100k provides insight into what is being blocked in each country, without introducing the biases of the people manually curating lists, such as the CLBL. In Table 2.6, we analyze the domains blocked in our Alexa experiment that were not included in the Citizen Lab experiment. Our domain categorization is performed by FortiGuard Labs, a common DPI tool provider, using their web interface [76].

Many of the domains we discover as blocked in our test of domains from Alexa are pornography. Interestingly, some domain classifications were not at all present in the Citizen Lab experiment, such as Shopping. Other categories, such as Personal Websites and Blogs and News and Media, can be extremely informative when considering what content is deliberately blocked by countries. Overall, we see that $3,130$ of the domains we observe as blocked are not in the CLBL. This is a significant improvement in coverage of blocked URLs, as we only see 220 URLs blocked from the Citizen Lab list.

Using the large set of domains tested, we can compare what domains are blocked in multiple countries, despite the sparseness of block list intersections. Many categories have domains that are not blocked in multiple countries, e.g. News and Media, meaning that the particular news sites blocked by each country are not the same as in other countries that also censor News and Media sites. In contrast, the set of blocked domains depicting violence and advocating extremism are the same in every country that blocks that type of content.

Finally, we utilize the ordered nature of the Alexa top domains to compare how each country's blocking changes with the popularity of a site, shown in Figure 2.9. While some countries show generally uniform distribution of blocking across the top $100,000$ domains, others show a tendency to select domains from the most popular. Countries demonstrating the tendency to block popular

domains with greater frequency are China, Egypt, and Turkey, with the strongest trend being that of Turkey. This may reflect a reactive blocking strategy, in which domains are added to a blacklist when they are detected to be visited with some frequency by citizens.

While the Alexa Top 100k experiment is only one snapshot of the state of application-layer censorship taking place on HTTP and HTTPS, we believe that it demonstrates the flexibility and accuracy of our tool. In the future, it can be used to contribute valuable data to many diverse, longitudinal, and in-depth studies of application-layer censorship.

2.6 Discussion

This paper has proposed and validated a technique for measuring application-layer interference around the world. In this section, we discuss the limitations of the design and what additional research our tool enables.

2.6.1 Ethics

Active network measurement [160], and active measurement of censorship in particular [112], raise important ethical considerations. Due to the sensitive nature of such research, we approached our institution's IRB for guidance. The IRB determined that the study fell outside its purview, as it did not involve human subjects or their personally identifiable data. Nevertheless, we attempted to carefully consider ethical questions raised in our work, guided by the principles in the Belmont [151] and Menlo [50] reports and other sources. We discussed the study's design and potential risks with colleagues at our institution and externally, and we attempted to follow or exceed prevailing norms for risk reduction in censorship measurement research.

Like most existing censorship measurement techniques, ours involves causing hosts within censored countries to transmit data in an attempt to trigger observable side-effects from the censorship infrastructure. This creates a potential risk that users who control these hosts could suffer retribution from local authorities. There is no documented case of such a user being implicated in a crime due to any remote Internet measurement research, but we nonetheless designed our technique and experiments so as to reduce this hypothetical risk.

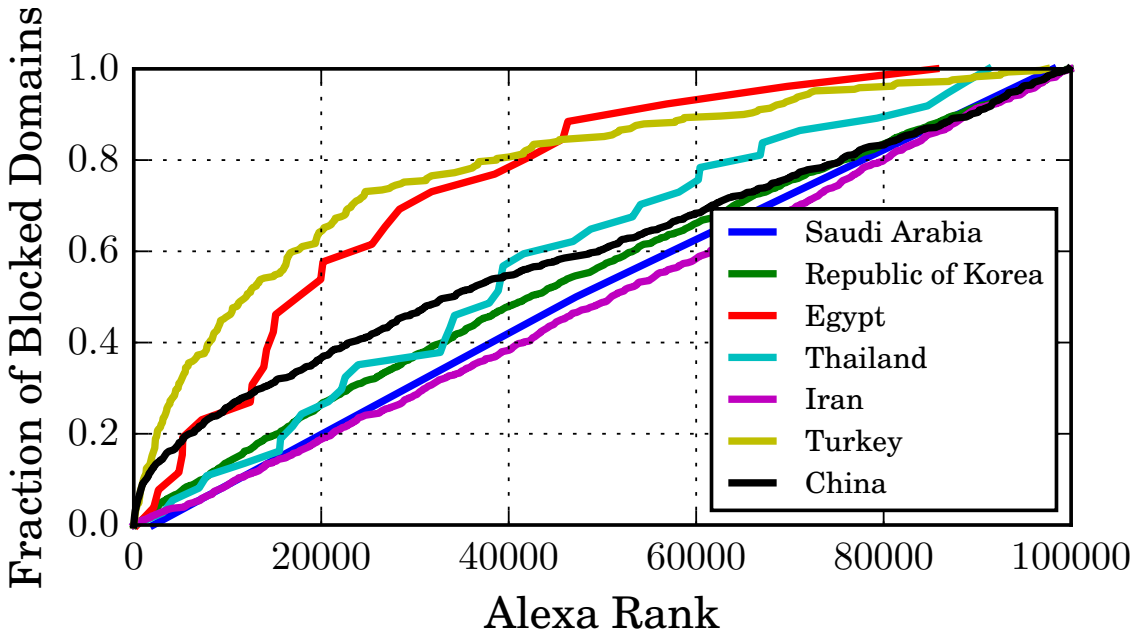


Figure 2.9: **Blocking by Alexa Rank**—The distributions of blocked domains relative to their Alexa rank varies by country. Egypt, Turkey, and China demonstrate a clear trend of blocking lower-ranked domains at a higher frequency. In contrast, Iran has a near uniform distribution of blocking across Alexa ranks.

Existing techniques [30, 161, 162, 180] in censorship measurement cause oblivious hosts in censored countries to make requests for or exchange packets with prohibited sites. In contrast, our measurements only involve connections between a machine we control and echo servers, so the echo servers never send or receive data from a censored destination.

Still, our interactions with the echo servers are designed to trigger the censorship system, as if a request for a prohibited site had been made. We cannot entirely exclude the possibility that authorities will interpret our connections as user-originated web requests, either mistakenly or by malicious intent. However, we believe that the actual risk is extremely small, for several reasons.

First, even upon casual inspection, the network traffic looks very different from a real connection from the host running the echo server to a prohibited web server. The TCP connection is initiated by us, not from the echo server. Our source port is in the ephemeral range, and the echo server's is the well known port 7. The first data is an HTTP request from us, followed by the same data echoed by the server, and there is never any HTTP response. The request itself is minimal, with no

optional headers, unlike requests from any popular browser. Any of these factors would be enough to distinguish a packet capture of our probes from real web browsing.

Second, the network infrastructure from which we source our probes looks very different from prohibited web servers. We tried to make it easy for anyone investigating our IP addresses to determine that they were part of a measurement research experiment. We set up reverse DNS records, WHOIS records, and a web page served from port 80 on each IP address, all indicating that the hosts were part of an Internet measurement research project based at our university.

Third, most echo servers look very different from end-user devices. We find (see Section 2.4.1.3) that the vast majority of public echo servers appear to be servers, routers, or other embedded devices. In the unlikely event that authorities decided to track down these hosts, it would be obvious that users were not running browsers on them.

There are additional steps that we did not take for this initial study that could further reduce the risk of misidentification. We recommend that anyone applying our techniques for longitudinal data collection incorporate them. Although we established that few echo servers are end-user devices by random sampling, in a long-term study, each server should be individually profiled, using tools such as Nmap, to exclude all those that are not clearly servers, routes, or embedded devices. In addition, the requests sent to echo servers could include an HTTP header that explains they are part of a global measurement study. This would provide one more way for authorities to conclude that the traffic did not originate from an end user.

Given these factors, we believe that the risks of our work to echo server operators are extremely small. We considered seeking informed consent from them anyway, but we rejected this route for several reasons.[1] First, the risk to these users is low, but if we were to contact them to seek consent, this interaction with foreign censorship researchers would *in and of itself* carry a small risk of drawing negative attention from the authorities. Second, if we only used servers for which the operators granted consent, these operators would face a much higher risk of reprisal, since their

[1] As discussed by others [160, 176], informed consent is not an absolute requirement for ethical research, so long as the research abides by other principles, e.g. those in the Belmont and Menlo reports or those steps proposed by Partridge and Allman [160], as we have strived to do.

participation would be easy to observe and would imply knowing complicity. Third, obtaining consent would be infeasible in most cases, due to the difficulty of identifying and contacting the server operators; if we limited our study to echo servers for which we could find owner contact information, this would lead to far fewer usable servers, thus severely reducing the benefit of the study. The communities that stand to benefit most from our results are those living in regions that practice aggressive censorship, and thus those who will likely benefit include the echo server operators in these regions, conforming with Menlo's Principle of Justice [50].

Beyond these risks, we also sought to minimize the potential financial and performance burden on echo server operators. We rate-limited our measurements to one concurrent connection per server, and each connection sent an average of only two packets per second. Our ZMap scans were conducted following the ethical guidelines proposed by Durumeric et al. [59], such as respecting an IP blacklist shared with other scanning research conducted at our institution and including simple ways for packet recipients to opt out of future probes.

We contrast our work with Encore [30], a censorship measurement system that has been widely criticized on ethical grounds. Websites install Encore by embedding a sequence of JavaScript. When users visit these sites, their browsers make background HTTP requests to censored domains, possibly without notice or consent. While we too make oblivious use of existing hosts without obtaining consent, the network traffic and endpoints differ dramatically from normal requests for censored content. We believe this substantially reduces the risk of harm.

2.6.2 Limitations

Our system currently relies on echo servers to gain perspective into remote client experiences of the Internet. Existing remote measurement techniques can be detected and invalidated or blocked by middleboxes, and ours is no exception.

First, the censor could block all traffic through port 7. We have no information about who or what else might be using port 7 today, so we have very little idea of how much collateral damage blocking port 7 would cause. Fortunately, our system is not dependent on using the Echo Protocol specifically; there are several other protocols that offer an echo service, such as FTP, Telnet, and

TLS. These other protocols would be much more difficult to block entirely, as they are used much more widely on the Internet. Many of these alternates do have the disadvantage of requiring a protocol-specific header, which may cause some middleboxes to stop responding to our probes.

Second, the censor could block our measurement machine by IP. One of the greatest advantages of our system is that it is portable; the measurements can be run from virtually any machine around the world. This means that any IP-based filtering of our measurements would likely be unsuccessful, as we could quickly and easily deploy in another location.

Finally, a censor could watch for the direction a connection and block only connections originating from inside their network. However, such a policy would not prevent services pushing data to clients, as can occur in FTP. In practice, we are not aware of directional blocking of this nature, potentially because the complexity of AS peering blurs the distinction of internal and external networks at a nation-state level.

However, both distributed and remote censorship measurement systems in use today are differentiable and disruptable. Even if some censors decide to disrupt measurements, we will continue to have visibility into the rest of the world.

Another limitation is the difficulty of detecting countries with heterogeneous deployments of keyword blocking, because in this work we considered only widespread blocking. Future work can remove our final Classifying Interference step, and instead combine the raw data with that of other network disruption measurement techniques [161, 162] to increase the granularity of observations.

Another limitation of the measurements conducted in this study, but not to our technique in general is that we have false negatives where DPI boxes monitor only port 80 and port 443 for web traffic. We could have conducted all of our experiments with our client port set to the appropriate well-known port for the protocol we would measure; however, we believed the trade-off was best to follow the best common practice and use an ephemeral port for our client connections.

One consideration in using this work for global detection is that there are only on average 177 countries with echo servers, and only 74 with at least 15 vantage points. One potential way to increase the number of vantage points is to send our formatted requests to any server that

accepts packets. For example, this could be done for HTTP by using all web servers. Then we would differentiate between the web server's error result and the interference behavior by country. However, this removes our ability to detect disruption that only inspects outbound packets from the network. Based on what we have observed in Section 2.5.3, this is a significant number of the countries that perform application-layer interference.

Finally, our work makes a trade-off to detecting censorship that is observed in multiple vantage points within each country, but this comes at the price of reduced granularity of observation. This means we will not regularly observe censorship that is heterogeneously implemented within a given country, and will not be able to reliably observe particular ISP policies.

2.6.3 Future Work

This paper describes a new and useful technique that can be used to remotely measure network disruptions due to application-layer blocking. Disruption detection techniques can monitor DNS poisoning, IP-based blocking, and now application-layer censorship. When combined, these perspectives could produce valuable datasets for political scientists, activists, and other members of the Internet freedom community. Additionally, these remote measurement techniques complement in-country probes, such as OONI, in order to provide baselines and focus effort.

The system presented here is capable of continuous measurement. Rather than regularly running a large batch of keywords, such as the Alexa list, a different optimization would cycle through a set of interesting domains in each country at a reduced rate. This would enable longitudinal tracking of those domains, and help illuminate how and when application-layer censorship policies change.

Quack also stands to provide interesting insight into censorship of other application-layer data and can be generalized to use other protocols' echo behavior. While we only focus on HTTP and HTTPS in this chapter, the Echo protocol's ability to send and receive arbitrary data could be used to explore interference in other areas, such as the mobile web and app ecosystems. Additionally, future work can be performed to use protocols other than the echo protocol. This would improve coverage of application-layer blocking measurement.

2.7 Conclusion

Application-layer interference is broadly deployed today, critically limiting Internet freedom. Unlike other techniques for censorship, we have not previously had broad and detailed visibility into its deployment. In this chapter, we introduced Quack, a new system for remotely detecting application-layer interference at global scale, utilizing servers already deployed on the Internet, without the need to enlist volunteers to run network probes. We hope that this new approach will help close an important gap in censorship monitoring and move us closer to having transparency and accountability for network interference worldwide.

CHAPTER III

Measurement in Aid of Censorship Circumvention

Measuring censorship is a useful tool in providing transparency to censored users about the system they have been forced into. However, it often is insufficient. In order to learn about the censorship practices of their country, a user must typically circumvent that censorship. This lands us in the business of providing censorship circumvention in our effort to provide transparency to users under censorship.

In particular, we deployed new censorship circumvention techniques to over 500,000 users in 12 different censoring countries over the last 19 months. This is the first deployment of Refraction Networking to real users. Our deployment has proved its utility to our deployment partner Psiphon [167], a circumvention client, keeping users on the free and open Internet even in times of heightened censorship. During our peak, we observed 40% of the traffic from the subset of Psiphon users we deployed to.

This chapter is adapted from two joint publications: the first appeared in the Proceedings of the 7th USENIX Workshop on Free and Open Communications on the Internet (FOCI'17) [79] and the second appeared in the 2020 Volume, 4th Issue of Proceedings on Privacy Enhancing Technologies (PoPET) [199].

3.1 Introduction

National-governments are deploying increasingly sophisticated systems for Internet censorship, which often take the form of deep-packet inspection (DPI) middle-boxes located at network choke-

points [51]. At the same time, popular circumvention techniques such domain fronting and VPNs are becoming harder to deploy or more frequently blocked [116]. There is an urgent need to field more advanced circumvention technologies in order to level the playing field.

One proposed new circumvention approach, Refraction Networking, has the potential to fill this need, and has been developed in the form of several proposed protocols [26, 66, 80, 101, 118, 135, 149, 210, 211] and other research [27, 87, 103, 148, 179] over the past decade. It works by deploying technology at ISPs or other network operators that observes connections in transit and provides censorship circumvention functionality. Though promising in concept, deploying Refraction Networking in the real world has faced a number of obstacles, including the complexity of the technology and the need to attract cooperation from ISPs. No Refraction implementation has ever served real users at ISP scale, leaving the approach's practical feasibility unproven before our work.

In this chapter, we describe lessons and results from a real-world deployment of Refraction Networking that we have operated in production for over a year and that is enabled in circumvention client software installed by more than 559,000 users. This began as a 2017 pilot, based on a high-performance implementation of the TapDance protocol [79]. It operates from stations installed at a mid-sized ISP that observe 140 Gbps of aggregate commodity traffic. We deploy at four network locations that individually process between a peak of 1–40 Gbps of commodity traffic for a total of 140 Gbps capacity, of which we observe a 70 Gbps aggregate peak. Of this, 500 Mbps was the peak throughput of circumvention traffic proxied by our system.

Operation of our deployment required solving technical, operational, and logistical challenges and necessitated close collaboration among researchers, network engineers, and circumvention tool developers. This reflects a non-trivial challenge to Refraction Networking systems: Refraction Networking does not function in isolation of the rest of the Internet, and instead must have an integrated role as a citizen of the Internet to be effective.

To meet requirements of our partnerships and facilitate further partnerships, we set the following system constraints:

- Utilize only 1U of rack space at each network tap location.
- Coordinate multiple stations as a single system.
- Operate continuously, without disturbing our partner ISPs' operations, despite individual server downtime in production.
- Continuously monitor our partner networks for decoy sites.
- Provide usable network performance to end-users in censored countries.

In this chapter we describe our experience meeting these constriants and the implications this has on the deplotment of Refraction Networking.

In addition, we analyze data from four months of operations to evaluate the system's performance. This four-month period reflects typical behavior for our ISP partners and concludes with a significant censorship event that applies stress to infrastructure of our circumvention tool deployment partner. It shows that our load is affected by censorship practices, and that we are able to handle the spike in utilization during a censorship event. During that censorship event we provide Internet access to more users than ever. And in doing so, we do not impose enough burden to receive additional opt-out requests from decoys, nor do we provide degraded quality of service.

Our final contribution is a discussion of lessons we learned from building and operating the deployment that can inform future work on Refraction Networking systems and other circumvention technology. In particular, decoy site discovery and reducing station code complexity are two main areas where future work would be greatly beneficial. We conclude that Refraction Networking can be deployed continuously to end-users with sufficient network operator buy-in, although attracting ISP partnership remains the largest hurdle to the technology's practical scalability. However, even with relatively limited scale, Refraction can meet a critical real-world need for a fall-back transport that can provide service when other, lighter-weight transports are disrupted by censors.

3.2 Background

Refraction Networking (previously known as "Refraction Networking") is a censorship circumvention strategy that places proxies at Internet service providers (ISPs) rather than at network endpoints. Users access these proxies by making innocuous-seeming connections to *decoy* websites, which pass by the ISP proxy station. The users signal the ISP proxy station by including a covert steganographic tag that only the proxy can see using a private key. The ISP proxy then impersonates the decoy server, and redirects the client's connection to the desired destination. To the censor, this connection looks like a normal connection to an unblocked decoy site. Since the decoy sites are not explicitly part of the proxy system, censors have a difficult time blocking them without blocking legitimate traffic to them as well.

In this section, we detail prior Refraction Networking schemes and provide some technical details on TapDance, the scheme we deploy.

3.2.1 Prior schemes

Refraction Networking was first proposed in 2011 with Telex [211], Curveball [118] and Cirripede [101], which proposed the idea of placing proxies at ISPs, with several variants on how clients could signal the ISP station. For instance, Curveball used a pre-shared secret between client and station, while Telex and Cirripede client's used a ISP proxy's public key to create steganographic tags, and embed them in either TLS client hello messages or TCP initial sequence numbers. Since both are typically random, without the private key, censors cannot distinguish these tags in proxy connections from the typical random values in normal connections.

However, all of these first-generation schemes require *inline blocking* at the ISP proxy, allowing the proxy to stop packets in individual tag-carrying TCP connections. This lets the proxy pretend to be the decoy server without the real decoy responding to the client's packets. While this makes for a conceptually simpler design, inline blocking is expensive to do in production ISP networks, where traffic can exceed 10s to 100s of Gbps. Inline blocking devices may be prone to failures, and ISPs were generally unwilling to suffer this risk to deploy Telex-era proxies.

To address this concern, researchers developed TapDance [210], which only requires a passive tap at the ISP proxy, obviating the need for inline blocking. This type of proxy is much easier for ISPs to deploy, since there is no longer a risk to impacting production (non-proxy) traffic if the proxy fails. We give a brief technical overview of how TapDance accomplishes this next in Section 3.2.1.1.

Beside TapDance, other Refraction schemes have proposed ways to make it harder to detect or fingerprint proxy connections. Slitheen [26] is a Refraction-style proxy that perfectly mimics the timing and packet size characteristics of the decoy, making it harder for censors to block based on web fingerprinting or other classification techniques. Both Rebound [66] and Waterfall [150] suggest ways to reflect client requests off decoys in order to enable ISP proxies that only see downstream (server to client) traffic. MultiFlow [135] provides ideas to adapt Refraction protocols to use TLS 1.3 [171], as most use prior versions and will need to be updated once older protocols are disabled. Finally, Conjure [80] is a recent Refraction protocol that utilizes unused IP addresses that go past the IP proxy to deploy *phantom hosts*. Since these phantom hosts are more numerous and cheaper to setup than real endpoint IPs, censors may find them more difficult to distinguish from legitimate hosts and block, particularly if each is only used ephemerally by a single client.

3.2.1.1 TapDance

TapDance only requires a passive network tap at the deploying ISP. Instead of inline blocking, TapDance clients mute the real decoy server in the connection by sending an incomplete HTTP request inside the encrypted TLS connection. The decoy server continues to wait for additional data, and meanwhile the ISP proxy pretends to respond as the decoy server. To the censor this looks like a complete (encrypted) request and subsequent response, and this allows the client and ISP proxy to communicate bidirectionally.

There are two main limitations with TapDance's technique. First, if the connection stays open too long, the real decoy might timeout and close the connection. Since its TCP state will be out-of-sync with the ISP proxy and client's state, this will appear as an obvious signal to the censor, allowing them to block the client's future connections. Second, the client cannot send more than a

TCP window of data to the decoy, since after this the real decoy will start responding with stale acknowledgments.

These behaviors limits the amount of data that a client can send in and the duration of a single TapDance connection. We overcome this by multiplexing long-lived (indefinite) sessions over multiple short-lived connections, which we describe in detail in Section 3.3.2.

3.3 Deployment Architecture

In this section, we describe the high-level architecture of our deployment, including the TapDance station deployed at a mid-sized ISP, and our client integration with a popular censorship circumvention app.

To distribute our client, we partnered with a popular censorship circumvention app developer, Psiphon , that has millions of Android and Windows users living in censored countries around the world. We integrated our client into their Android and Windows' client, and pushed it to a cohort of 559,000 users in 9 censored regions.

3.3.1 ISP Deployment

We deployed TapDance in a mid-sized regional ISP, Merit Network, that serves several universities and K-12 institutions in North America. We collect up to 140 Gbps of production traffic from four vantage points within this ISP, allowing us to observe thousands of websites that can serve as potential decoys for proxying.

We note that our four network vantage points do not cover every entrance to the ISP's network, and so there are paths into the network that do not pass any of our packet taps. By traffic volume, our taps observe approximately 80% of the packets entering the ISP's network.

3.3.2 Long-lived Sessions

The TapDance protocol imposes several limits on client-to-decoy connections. For instance, connections can only be maintained until the decoy decides to timeout the connection (as it hasn't received client data at the application level). While we select decoys that have longer timeouts, this is nonetheless on the order of 20–120 seconds. Another limit is the upload limit, in which clients

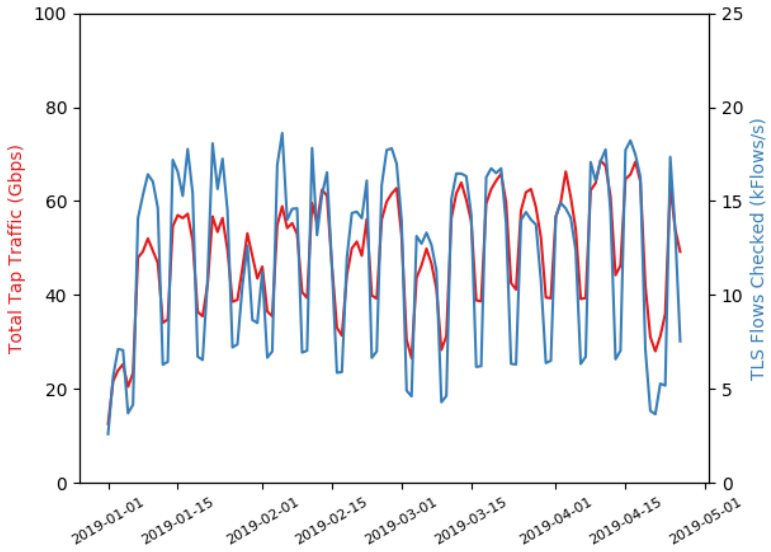


Figure 3.1: **Tap Traffic**— Our taps have capacity for a peak 100 Gbps of mirrored traffic from four tap locations. This peaked under 70 Gbps during our measurements. This corresponded to between 5,000 and 20,000 TLS flows per second during a typical week. There is a weekly pattern of low traffic on weekends. Periods with traffic less than 40 Gbps longer than two days correspond to start and end of semesters at the institutions we our parnter ISPs serve.

sending more data than the decoy server's TCP window to the decoy will cause the decoy to start responding with ACK packets. If a client wants to have a connection live for longer than the timeout or be allowed to send more than the upload limit, they must multiplex their connection over multiple connections to the decoy.

We term a single connection to a decoy as a **TapDance flow**, and the longer-lived underlying connection to a particular covert address (e.g. a blocked website or proxy) as a **TapDance session**. Clients choose a random 16-byte session identifier, which is sent in the first TapDance flow to the decoy site. On the station, this first connection sets up the session state and connects the client to the covert address. Before the flow times out or the client sends beyond the upload limit, the client closes the flow, and opens a new one with the same session identifier. The station then connects the new flow to the previous session, giving the client the appearance of an uninterrupted long-lived session to the covert address.

3.3.3 Handling 40Gbps Links

We reimplemented the TapDance station from scratch, primarily in Rust. Rust is an emerging low-level systems language that provides compile-time guarantees about memory safety, which lowers the risk of remote compromise. To efficiently process packets at 40 Gbps line rates, our implementation is built on the PF_RING library and kernel driver [155], operating in zero-copy mode. By splitting incoming traffic onto multiple cores we were able to handle full line rate traffic with only occasional dropped packets, which, due to the design of TapDance, do not interfere with the normal operation of an ISP. Our experience demonstrates that, even in large installations, a software-based implementation of TapDance can be practical, avoiding the need for costly specialized hardware.

The station uses C in three main areas. First, it uses a C implementation of the Elligator [24] tagging scheme for better performance during initial detection of TapDance flows. Next, it uses OpenSSL to handle TLS work, including manually assembling TLS connections from secrets extracted from TapDance tags. Finally, it uses `forge_socket`, a Linux kernel module that allows a userspace application to create a network socket with arbitrary TCP state, in order to utilize Linux's TCP implementation while pretending to be the server.

Handling multiple TapDance users is highly parallelizable, so the station is designed to use multiple cores by running multiple processes with absolutely no communication among them. However, a single TapDance session typically spans multiple TLS flows. This means that, once a user has begun a TapDance session on a particular station process, that same process must see all future TLS flows in that session. To achieve this, we configure PF_RING to spread flows based on the IP pair, rather than the full TCP 4-tuple, so that the client's changing source port will not send the traffic to a different process.

Our four stations ran on a total of 34 cores (excluding a dedicated PF_RING core per station), with the most loaded station using 14 cores. These 34 cores were able to comfortably handle a peak of close to 14,000 new TLS connections per second during our trial, with each connection being checked for a TapDance-tagged request.

3.3.4 Multi-Station Constellation

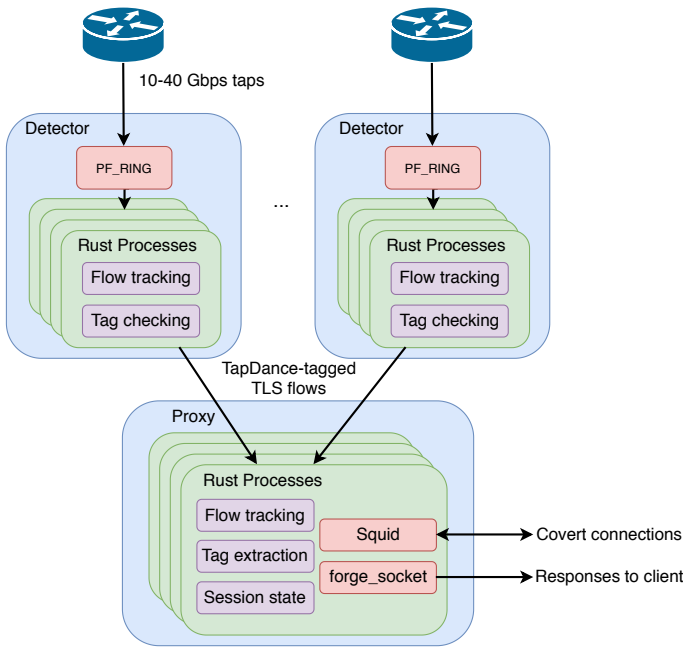


Figure 3.2: **TapDance Station Architecture**—Multiple detectors, central proxy

We split our station into two components: multiple **detectors** and a single central **proxy**. Detectors process raw traffic from packet taps, and look for tagged TapDance TLS flows. When a tagged flow is identified, its packets are forwarded using ZeroMQ [98] to the central proxy running elsewhere in the ISP. This architecture allows clients to use multiple different decoy servers within the same session, as their session state is maintained by the single central proxy that the detectors forward to. Figure 3.2 shows a high-level overview of the TapDance station architecture.

Detectors use PF_RING [155] to read packets from the ISP taps, which range from 10 to 40 Gbps. PF_RING allows us to split packet processing across multiple (4–6) cores on a detector, while ensuring that all the packets in a specific flow are processed by a single core, reducing the need for inter-core communication. Detectors perform flow tracking of TLS flows, and perform a cryptographic tag check using Elligator [24] on the first TLS application data packet in a flow. Typically, detector stations process anywhere from 300 to over 16k new TLS flows per second.

Once a TapDance-tagged flow is observed on a detector, it forwards the flow's TCP SYN packet, the tag-carrying application data packet, and any subsequent packets in this flow to the central

proxy. The proxy thus only receives packets for flows that are related to TapDance connections. The central proxy also contains multiple processes, and ZMQ forwards TapDance flows based on their session identifier, allowing all of a sessions' flows to be sent to the same core on the central proxy, reducing the central proxy need for inter-process communication.

For each TapDance session, the central proxy maintains a connection to a local Squid proxy, which the client uses to connect to covert destinations. The central proxy also uses a custom Linux kernel module named forge_socket that is used to establish sockets with arbitrary IP/TCP parameters. forge_socket is used to provide the station with a socket that impersonates the decoy server in a TapDance flow. This lets the central proxy call `send` and `recv` on the socket to produce and consume packets in the TapDance flow, as if it were the decoy server.

One limitation of this architecture is that packets are received from the client in a different network location (at the detector) than they are sent to the client (at the central proxy). This could potentially be used by censors to infer the presence of TapDance, either via observing TTL, timing, or network arrival point differences. Although we do not see evidence of censors using this (or any) technique to block TapDance over the year, one patch to this potential attack is to forward packets back to the detector corresponding to each TapDance flow, and injecting packets into the network there.

3.3.5 Client Integration

Regular users only access refraction networking through the Psiphon app. Psiphon is a censorship circumvention tool that uses several transports to provide uncensored Internet access to its users. We partner with Psiphon in a mutually beneficial manner: we do not need to invest in community building and app development to test our transport, and they get improved diversity of circumvention techniques. Psiphon uses several modular transports, automatically picking one of them for each user connection.

The Psiphon application selects the modular transport to use based on the time it took to fully establish the connection. When an uncensored tunnel is needed, Psiphon attempts to establish connections using all available transports. The transport, that successfully establishes the connection

first, is then used, while connections made by other transports are discarded. This selection algorithm provides optimal user experience by prioritizing working techniques with lowest latency.

Our technique is deployed as one of these modular transports for a subset of Psiphon users. We are able to deploy TapDance-enabled version of Psiphon to the users in particular countries and networks, but not to subsets thereof because Psiphon implements no tracking of individual users. We do not alter our deployment in the study period analyzed in this chapter, however, we have successfully deployed to users in several countries facing censorship.

Both our gotapdance library and Psiphon app are written in Golang, simplifying the integration. gotapdance library provides `Dialer` structure, that implements standard `net.Dialer` interface. This interface specifies `Dial` and `DialContext` functions to establish connections over TapDance to arbitrary addresses and return a TapDance connection in a form of a standard `net.Conn`. Implementation of standard interfaces simplifies the integration by providing familiar API, and improves modularity, allowing Psiphon to reuse existing code, that works with standard `net.Dialer` and `net.Conn` interfaces.

While establishing TapDance connections is easy with standard interface in gotapdance, there are 2 functions that Psiphon needs to call first. One of those functions is `gotapdance.EnableProxyProtocol`, and it will modify TapDance requests to ask the TapDance station to send the HAProxy PROXY [186] protocol header to the destination address before starting to tunnel the connection. We use PROXY protocol header because it includes the IP address of the client, which is important to share with Psiphon back-end to collect statistics and prevent abuse. Secondly, Psiphon needs to call the `gotapdance.AssetsSetDir` function to specify a writeable folder for assets, and allow TapDance library to persistently store the updates to the Client Configuration file, which includes the list of decoys.

To further improve integration through testing, Psiphon provided us with instructions on how to compile a version of their application, that only uses TapDance transport, so we can integrate this build with our Continuous Integration system. On any code change to the TapDance library, our CI would automatically run the tests, and build Android and Command Line Interface versions of

Psiphon application for manual testing.

3.3.6 Monitoring

To quantify the performance of TapDance we rely on a system of logging and analysis technologies that aggregate information from each individual station. Detectors are responsible for identifying tagged flows out of the full stream of traffic at any given tap and as such they track the number of packets checked, the traffic flow rates, and the current number of live sessions among other things. The proxy is responsible for traffic routing and session management, communicating directly with clients. The metrics that it produces allow us to track and associate flows with sessions and monitor the number, duration, throughput, and client round-trip time. These metrics quantify the performance, resilience, and (when any station fails) adaptability of the TapDance system as a whole.

The information produced by each stations is collected by Prometheus [166] and visualized using Grafana [88] to provide a real-time view of the health of TapDance. The system has also been instrumented to prevent overloading of decoys, and alert developers when an outage occurs. Logs are also backed up using an Elastic Stack [64] instance to allow for longer term aggregation and evaluation. This monitoring architecture allows the performance and reliability of TapDance to be quantified despite the distribution of responsibilities.

3.3.6.1 Station Placement and Decoy Discovery

Two important and related questions that a refraction networking deployment needs to address are where to place refraction detection stations and how the clients will choose which overt decoy destination to contact. The former question has largely been studied in the context of AS-level deployment to transit ISPs [37, 87, 102, 148, 179]. The latter has not been explored in detail, though Houmansadr, et al. [102] do consider how effective randomized client-side probing can be for a given level of deployment.

Our deployment has characteristics that are significantly different from those explored in previous work. Our partner ISP is a regional access ISP, acting as a primary point of Internet access

for its customers. At the same time, it has many peering and client connections at numerous points of presence, so capturing *all* traffic that transits our partner ISP is challenging. Additionally, many of its customers do have other network interconnections on which traffic arrives, complicating the choice of covert destinations.

3.3.7 Reachable site selection

The nature of our deployment meant that we can enumerate the customers that are served by our partner ISP and their corresponding AS numbers. This allowed us to enumerate all of the publicly reachable websites operated by our ISPs clients. This identifies roughly 3000 potential sites in our deployment. Not every site is usable with TapDance, however. There are four potential problems:

- Does the site have a large enough initial TCP send window to allow the sending a sufficient amount of data? A reachable site will ignore the acknowledgments for data that have been spoofed by a refraction station as long as they fall within its TCP send window; out-of-window acknowledgments, however, cause a TCP RST to be sent, breaking the connection and violating covertness.

- Does the site support a large enough timeout to keep a TCP connection open after a partial request has been sent? Again, a too short timeout will mean the site will try to close the connection while it is still being used for a refraction session, rather than remain silent.

- Does the site support and select a TLS cipher suite compatible with our TapDance implementation? Currently we support only AES GCM ciphers, and send a Chrome 62 TLS fingerprint. If the site selects a different cipher, we will be unable to send the TapDance tag, and therefore be unable to use the connection.

- Does traffic from a client to the reachable site pass a station?

To deal with the first 3 problems, we implemented a testing script that check the size of the TCP window, the duration of timeouts, and the selected TCP cipher suite. Figure 3.5 shows the results of our tests over the course of our test period, and this is discussed in pore detail below. Our initial

approach for the last problem was to simply rely on station placement to attempt to capture most traffic transiting our partner ISP, as discussed below.

3.3.7.1 Decoy Filtering

To find potentially suitable decoys, we first start with a list of all subnets within our partner ISPs' networks. We obtain a set of potential decoys by scanning port 443 across this address range, and noting the domain names on certificates returned. This provides a relatively small list of domains we know are served validly in the address ranges served by our partners, which is accumulated over our deployment.

These webservers are then filtered down first by identifying those that are can provide long connections to the client. We measure the TCP window size of the server to evaluate how much the client can upload before needing to a new connection to the decoy. Similarly, we measure the server HTTP timeout to evaluate how long the client can use a single connection. We filter out servers with TCP window less than 15 KB and timeout less than 30 seconds. Distributions of these parameters in a typical measurement are shown in Figure 3.4 and Figure 3.3.

Next, we apply a local blacklist of subnets we know have decoys that behave poorly when used as decoys. This can occur due to server reliability or throughput issues.

Next, we remove the domains that include our client in their `/robots.txt` file, as described in our opt-out instructions provided to all decoys via a link the user-agent sent by clients. As of early February 2020, only two domains have opted out, both before January 2019.

Finally, we perform a test connection with our client to ensure a correct cipher suite is chosen and that the connection is functional. This step removes the most decoys, due to cipher suite selection and detection station placement. We see major drops in the success rate of this step when we have misconfigurations of key infrastructure between the detectors and our discovery process. To protect from major changes affecting system availability, we do not deploy new decoy lists to clients if there a removal of larger than 30% of the previous day.

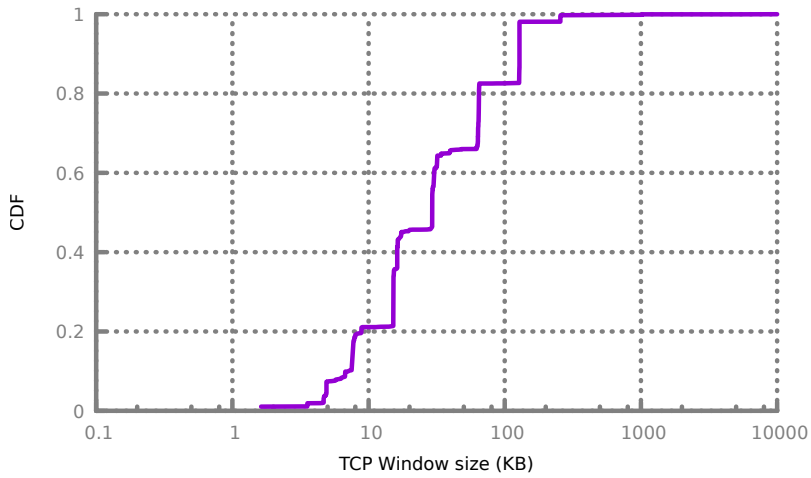


Figure 3.3: **TCP Window Size of Candidate Hosts**—We measured the TCP window size for each candidate reachable host, in order to find ones suitable for TapDance.

3.3.7.2 Detection Station Placement

Our partner ISP has numerous points of presence, routers, and peering points. Although previous work on Refraction Networking generally assumes that any traffic that transits an AS is observed by a detection station, in practice there are constraints that keep this from being true. As pointed out by Gosain, et al. [87], detection stations capable of processing high line rates are expensive, although our costs are at least an order of magnitude smaller than the estimates used by Gosain, et al. (which were aimed at Tier 1 ISP). There are additional costs, from finding rack space in the points of presence, to engineering time to deploy and manage the detectors. As a result, our deployment consisted of three detector stations initially, growing to 4 during our deployment. The fourth station was added when a change in routing caused a significant fraction of incoming traffic to arrive in a geographic location that did not have a detector. Three of our stations use 4x10 Gbps and one uses 2x10 Gbps, thus handling up to 140 Gbps of traffic. Our detectors monitor approximately 80% of all flows that arrive at our partner ISP.

3.3.8 Managing Load on Reachable Sites

In our tests with our own Nginx and Apache servers, we noticed that under default configurations, these web servers limited the number of simultaneous connections they would allow. For example,

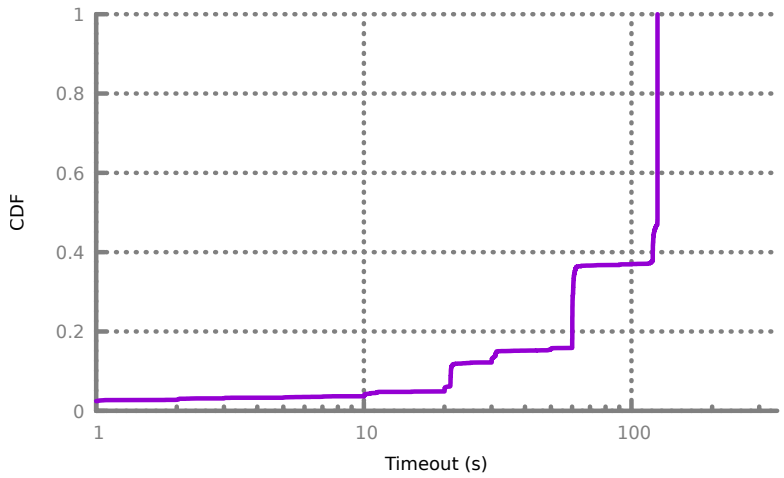


Figure 3.4: **Timeouts of Candidate Hosts**—We measured how long (up to a limit of 120 seconds) candidate reachable hosts would leave open an incomplete HTTP request.

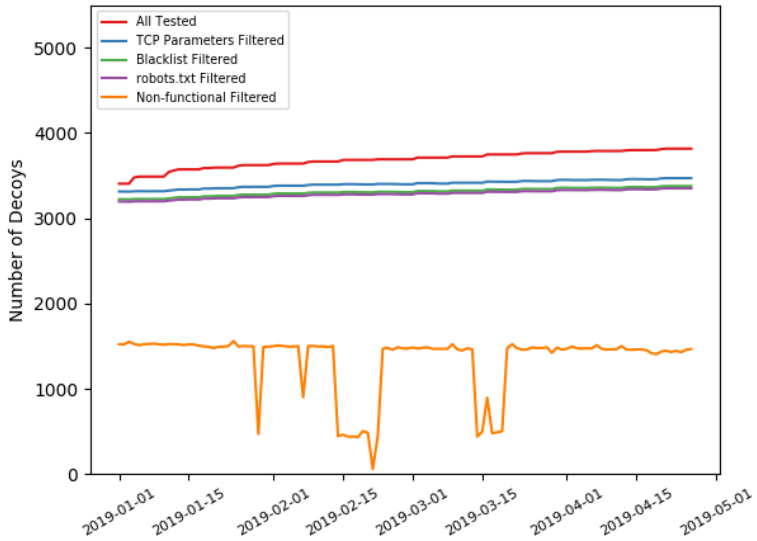


Figure 3.5: **Decoy Discovery Process**— Over the course of our study we attempted to discover new decoys each day. We accumulate all IP addresses that host a web server over our study period and test in four stages before they are published to clients. We observe two web servers asking us to not use them through their `/robots.txt` file. The step with the greatest loss is when we attempt to connect to them with a real client. There are also several troughs in this line that indicate outages in one of our detector sites that provided access to a majority of our decoys from the perspective of the server conducting discovery.

after 150 connections, a default Apache webserver will stop responding to new connections until one closes. Since a client will continue using a chosen reachable site for a long period, we were

concerned that "hot spots" could develop where particular hosts received disproportionate load.

We addressed this potential issue by attempting to limit the number of concurrent users to any individual site. Because users may reach a site through multiple stations, this requires coordination between stations to track how many active clients are using a particular site.

We added support in our stations to allow a central collector to inform a station that a particular site was overloaded and it should turn away additional clients attempting to use that site. We initially set the per-site limit to 30 concurrent connections to prevent inadvertent overload, and the site load remained well below our expected threshold for the duration of the trial.

In addition to this change, we also gave clients that failed to connect to a site an increasing timeout before trying the next site. This exponential back off ensured that, if a station or other failure occurred, the load clients pushed on sites or other network infrastructure would decrease rapidly.

3.4 Trial Results

We present an initial analysis of the results from our trial deployment in the spring of 2017.

3.4.1 At-Scale Operation

A major goal of this deployment was validating the hardware and operational requirements for running TapDance in a production setting. We were able to run our system while processing 40 Gbps of ISP traffic from commodity 1U servers. The observed traffic over our measurement week is shown in Figure 3.1, showing a cumulative processing peak of 55 Gbps across stations, and a single station processing more than 20 Gbps. During our trial, the CPU load on our stations remained below 25%, although this figure alone is not representative of achievable performance, which is heavily dependent on scheduling between the processors and network card.

3.4.2 User Traffic

Over the trial, we served over 50,000 unique users, according to Psiphon statistics. For privacy reasons, we did not track identifying information of users during our trial. Instead, Psiphon clients reported the last successful connection time, whenever they reconnected. This allowed Psiphon

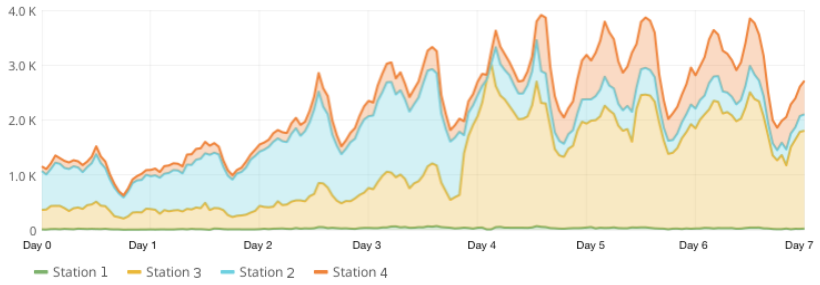


Figure 3.6: **Concurrent Sessions**—This plot shows the number of active TapDance user sessions, which peaked at 4,000.

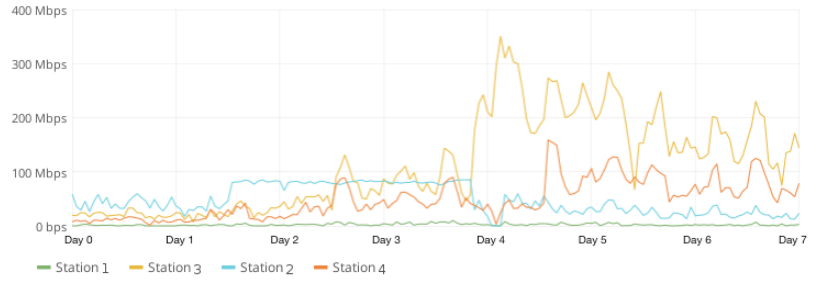


Figure 3.7: **User Traffic (Downstream)**—This plot shows how much user traffic our deployment served to clients. In response to the saturation of the 100 Mbps link on Station 2, we took steps on Day 3 to migrate user load to Station 3.

to query for connections with a last connect time before the current day to get a unique daily user count, as shown in Figure 3.13. Figure 3.6 shows the number of concurrently active users over the trial week. At peak, TapDance served over 4,000 users simultaneously, with peaks on a single station over 3,000 concurrent users. Interestingly, we see two peaks per day, which is not explained by user timezone distributions.

On day four of the trial, we discovered a bottleneck that was limiting use on the most popular station (Station 2). At this station, the injection interface was misconfigured to be over a 100 Mbps management interface. This link was saturated by the amount of traffic being injected by our station, making downloads noticeably slower for users of this station. The station was at a remote facility where it was impractical to reconfigure the network, so we worked with Merit Network to shift traffic headed to a large subset of reachable sites so that it passed a different station (Station 3) with a 1 Gbps injection interface. This solution improved download performance considerably, reflected in the observed usage in Figure 3.7.

Figure 3.8 shows a lower-bound estimate on the bandwidth available to users. The average throughput of 5 KB/s shown in this CDF does not reflect users' actual experience, as many sessions included in the CDF did not see real usage. The structure of our deployment was such that a users' entire device's internet traffic would be tunneled through a single, long-running TapDance session. These sessions must be occasionally restarted, due to network instability or other problems (see Figure 3.10). Therefore, many of the observed sessions happened entirely while the user was not using their device at all.

Figure 3.10 shows the distribution of client session durations. The median individual session length to a station was on the order of a few minutes. We reviewed logs to determine that about 20% of sessions ended in a timeout (where the client was not heard from for over 30 seconds).

3.4.3 Impact on Reachable Sites

During the trial, we measured the impact of multiple clients using the same site to reach TapDance. As described in Section 3.3.8, we are attentive to the number of clients simultaneously using a given site, as large numbers of clients could lead to resource exhaustion.

Figure 3.11 shows the load on the median, 90th, and 99th percentile sites over the trial. The median load remained generally evenly spread, with the typical site seeing around 5 clients connected simultaneously, with only 10% of sites ever having more than 20 simultaneous users.

Mid-week, we shifted more users toward Station 4 by increasing how often clients picked sites that were likely to be served by that station. We simultaneously increased the maximum simultaneous connections allowed to each site from 30 to 45. This is reflected in the 99th percentile site load, which peaked at 37 concurrent users to a single site.

3.5 Deployment Results

In this section we evaluate our deployment of TapDance. While our deployment is still proceeding for its 18th consecutive month, we present four months of data from the start of 2019. As our partner ISPs serve academic institutions, this is one typical semester: a high-load period for academic ISPs. This was also a period where our user-base was not altered by us, and no

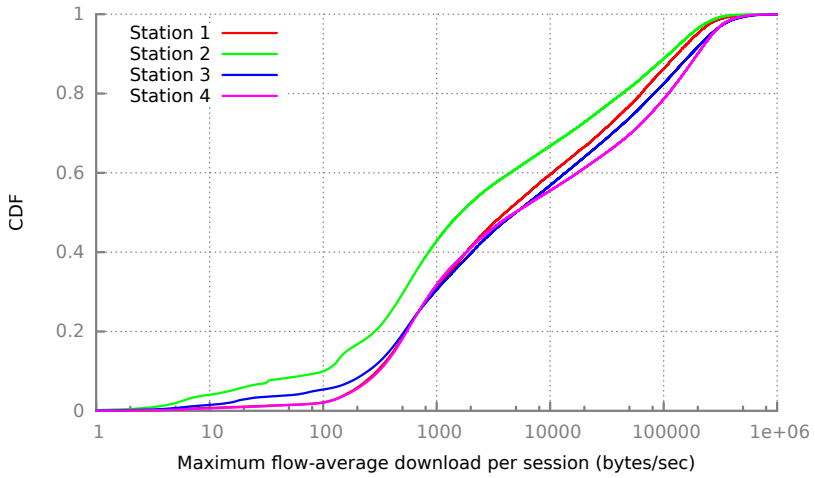


Figure 3.8: **Session Throughput**—This CDF shows the average downstream throughput achieved by the fastest sub-flow during each client session. Note that this is a lower-bound for actual throughput, as it is possible that sub-flows were not actively used for their entire duration.

major engineering changes were pursued. This allows a view of a steady-state of the aspects of the deployment within our control.

We observe changes in TapDance use over this period in spite of our consistency. These changes are due primarily to changes in censor behavior. A significant fraction of our usage occurred in a single censoring country, so actions taken to disrupt Internet freedom affect how circumvention protocols are used. The most salient feature reflecting this is the major censorship event in mid-April that caused TapDance usage to more than double.

We note that our evaluation period ends after traffic returns to normal in late-April. At this point, we restarted our collector station for maintenance, and inadvertently disabled much of the logging that we use for our analysis.

3.5.1 Psiphon Impact

During our observation period, TapDance was one of a handful of available circumvention strategies, and we served approximately 10% of traffic of Psiphon users that were TapDance-enabled. The number of daily users this reflected varied significantly in a weekly pattern, with between 5,000 and 15,000 users. Under significant censorship of other protocols, TapDance usage peaks above 40% of clients' traffic and 25,000 daily users, as shown in Figure 3.12 and Figure 3.9 respectively.

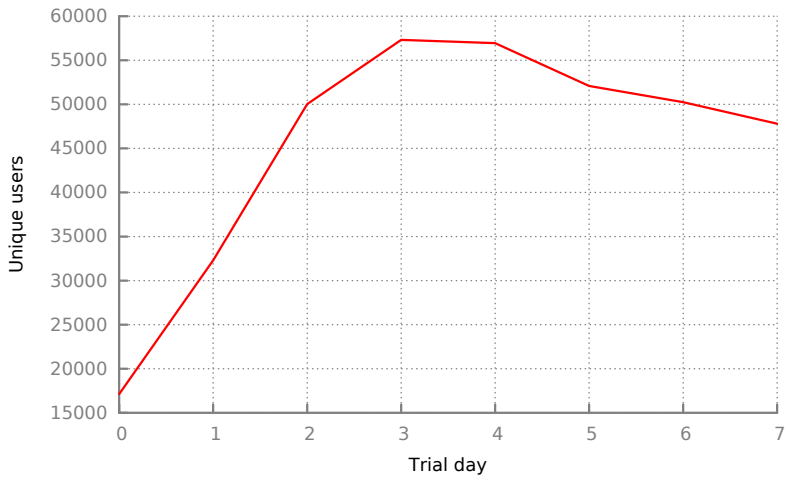


Figure 3.9: **Unique daily users**—The number of unique connected clients over each day of our measurement week. At peak, we handled over 57,000 unique users.

The particular behavior that caused this peak was censorship of other circumvention techniques that are typically very reliable. When this censorship occurs, clients will switch over to the remaining available transports, including ours, causing an apparent influx of TapDance clients. We also observe the opposite: when other low-latency circumvention protocols are temporarily unblocked, users will tend to select those, causing a decrease of TapDance clients, as seen in early March.

3.5.2 Client Performance

To demonstrate the utility of our service to clients, we evaluate their performance, the net throughput of the system, ensure no degradation of service under load, and ensure that no clients are monopolizing the system resources.

We do not track individual clients for their privacy. Instead, we log the subnet (/24) the client appears to connect from. Even at this coarser granularity, we do not observe dominance of any system resources by a set of clients, as shown in Figure 3.14. We analyze the number of bytes up, bytes down, and session duration, and find that these all correlate strongly, showing a similar use of bytes and duration per session across all client subnets (bytes up and down, r=0.88; bytes up and duration, r=0.82; bytes down and duration, r=0.74). This suggests that most users receive similar performance.

Over this study period, the average user data throughput did not drive or depend upon system

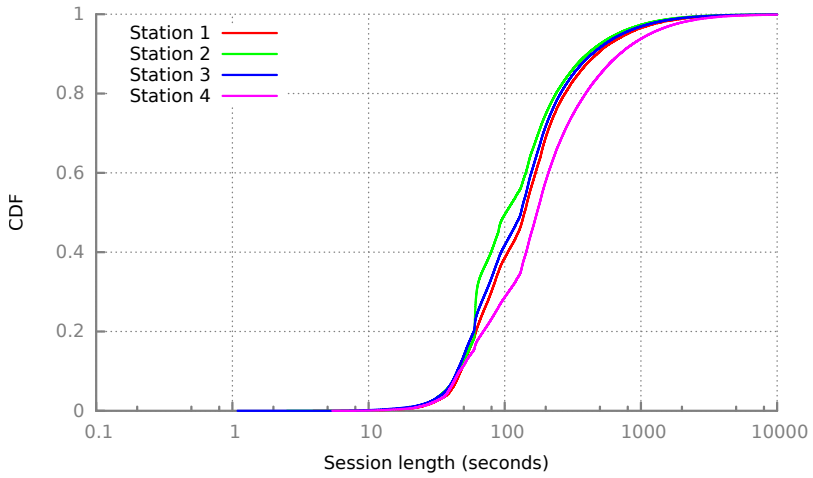


Figure 3.10: **Session Length**—This CDF shows client session durations over our trial, i.e., the time each client stayed continuously connected to a station without interruption. When sessions were interrupted (due to network failures or timeouts), the client automatically reconnected.

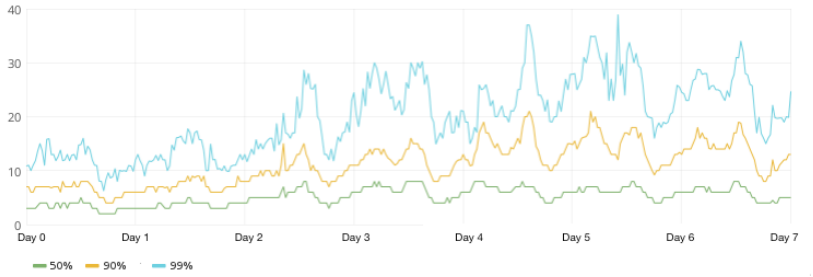


Figure 3.11: **Clients per Site**—This plot shows how many clients were connected to the average, 90th-, and 99th-percentile reachable site during our trial deployment.

utilization; rather, total system throughput depended upon the number of clients using the system, as shown in Figure 3.15. This indicates there was likely capacity during much of our study that went unused due to the protocol selection strategy of Psiphon .

Clients that are used to transmit data see an average round-trip-time (RTT) under 1 second for the entirety of our study. We note this may be due to survivorship bias in our measurement strategy, as clients that took longer may use another transport, closing connections before we receive their metrics. The time it takes to transfer the first byte of the connection is considerably longer, often over 5 seconds, as shown in Figure 3.16. This is due to the large number of round trips required before useful data is transmitted.

In spite of the long time to first byte, our clients observe reasonable session lengths in bytes

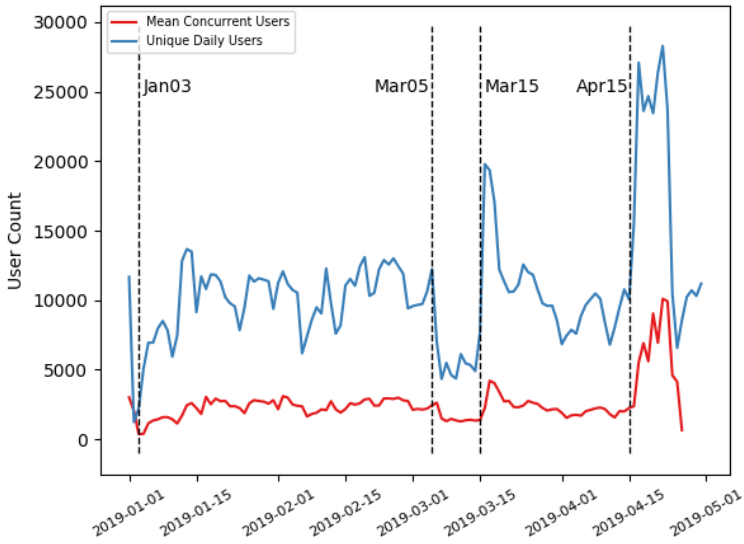


Figure 3.12: **User Counts**— Our user-base was composed of tens of thousands of users in censoring countries. We typically had approximately three thousand concurrent users and ten thousand daily unique users. However, this varied over our measurement. We indicate changes in observed censorship behavior that affects the country with our largest userbase on the graph. **Jan03**: Censors reduce restrictions on Domain Fronting. **Mar05**: Censors reduce restrictions on protocols over SSH. **Mar15**: Protocols over SSH and several new Domain Fronting providers are blocked. **Apr15**: New censorship capabilities demonstrated, restricting several long-reliable techniques.

and time, shown in Figure 3.17 and Figure 3.18 respectively. Client connections last approximately 90 seconds and transfer 20kB in a median session. Moreover, during the censorship event, under increased load, we do not observe degradation in these distributions.

3.5.3 Decoy Impact

One concern we and our ISP partners have is the extent to which our system burdens sites used as decoys. Of primary concern is ensuring that we do not overload any single decoy with a majority of our traffic or with a large numbers of concurrent connections. In Figure 3.19 we show that our impact on decoys is generally small. In Figure 3.20 we see that half of all decoys have two or fewer concurrent connections over our observation period.

In spite of the top 10% of decoys seeing the most use, the peak usage is restricted to below an average of 50 concurrent connections over the busiest day. Additionally, over the entire test period the busiest decoy saw an average 13.24 concurrent connections, and corresponding to 12.32 MB of

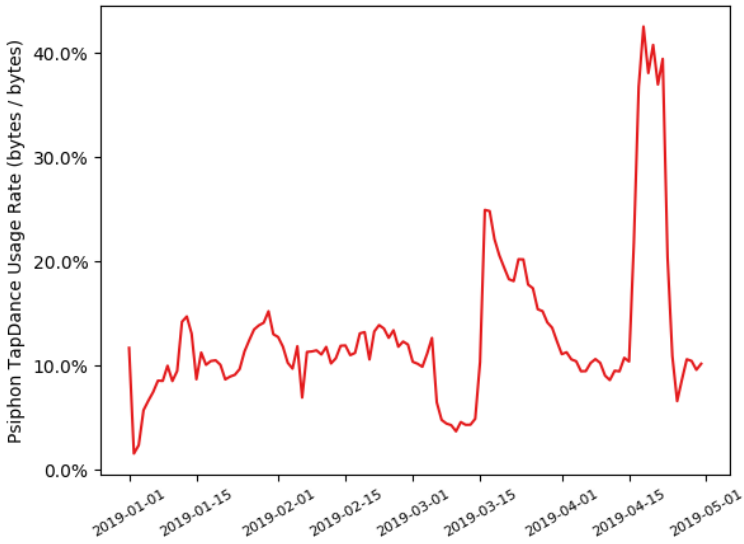


Figure 3.13: **Psiphon Tapdance Usage Rate**— We show what percent of bytes transmitted and received by users with TapDance as an option are transmitted through TapDance. This graph has the same features as those in Figure 3.12, indicating that our user-base size is driven by how frequently Psiphon users select TapDance.

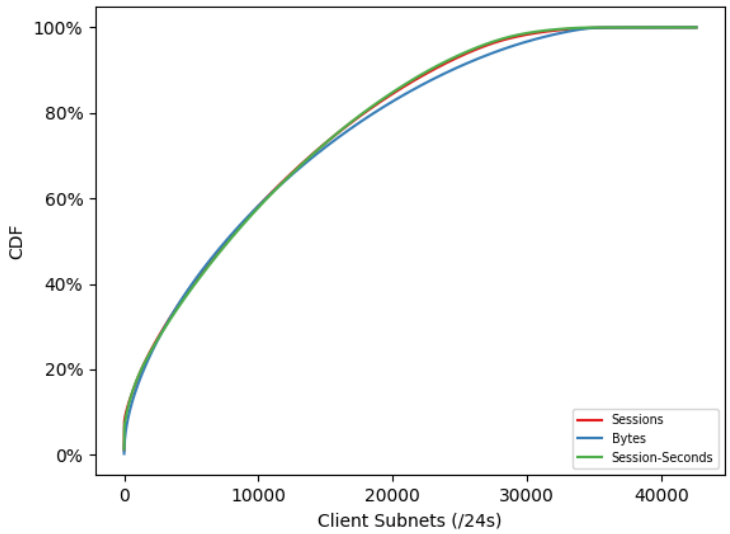


Figure 3.14: **Client Subnet Distribution**— We show the distribution of client consumption of sessions, data transmission, and decoy utilization over our measurement period. We see no major difference in any of these lines, indicating a uniform allocaiton of resources per session over our user-base. Half of all resources are being used by 8,000 client subnets and 1% of all resources are used by 10,000 client subnets.

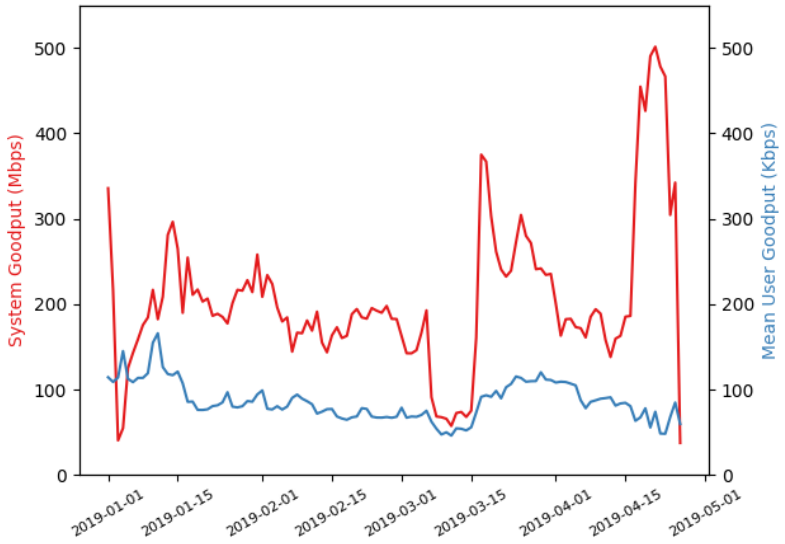


Figure 3.15: **Goodput**— Over our study period we provide useful data transmission that peaks around 500 Mbps during our last week. The user traffic per session corresponded to approximately 100 Kbps per user throughout the study period, and did not react negatively to incresed user load.

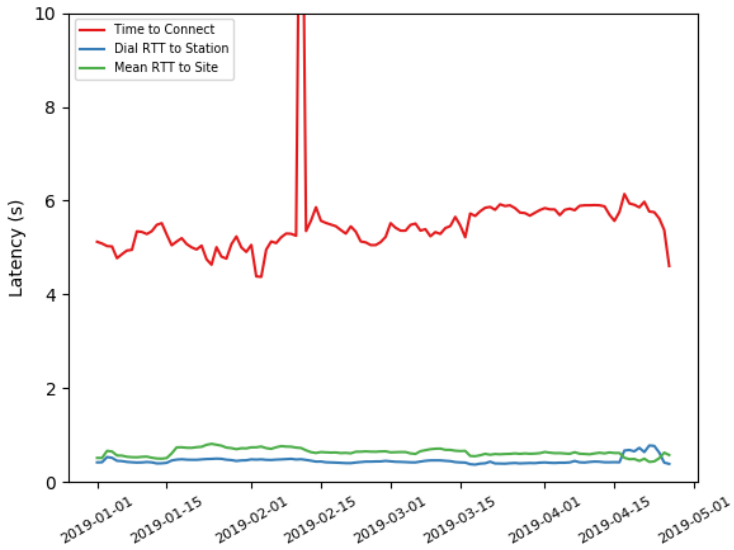


Figure 3.16: **System Latency**— Our system's latency is steady throughout our measurement period. The time to connect does peak in early February when a brief outage caused persistent clients to wait very long for the system to return to connect. We also note that there was an increase in dial RTT toward the end of our measurement. This is shown more clearly in Figure 3.27.

uncensored traffic per day, shown in Table 3.1. We note the decoy does not see the bulk of proxied traffic, though it does receive (and ignore) data and acknowledgements from the client.

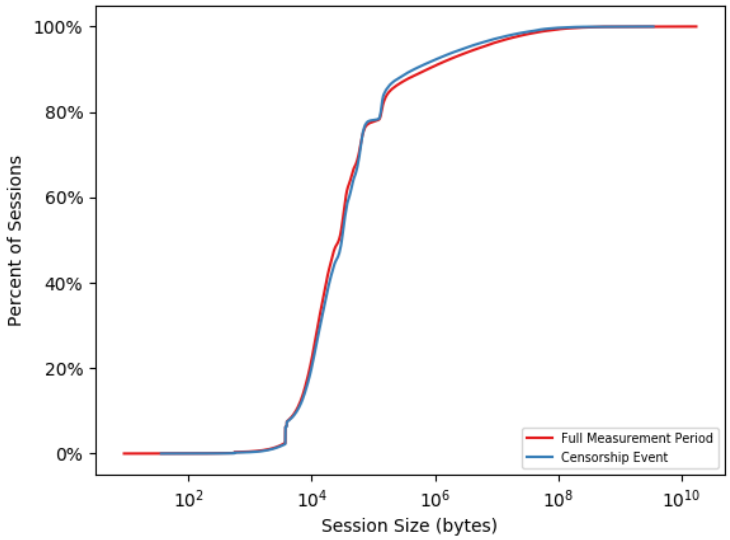


Figure 3.17: **Session Size Distribution**— Clients see a range of connection sizes over both our full measurement period and the censorship event in our last week. Over 90% of sessions transmit and receive over ten kilobytes. Moreover, we do not see a large effect on session size during the increased utilization of the censorship event.

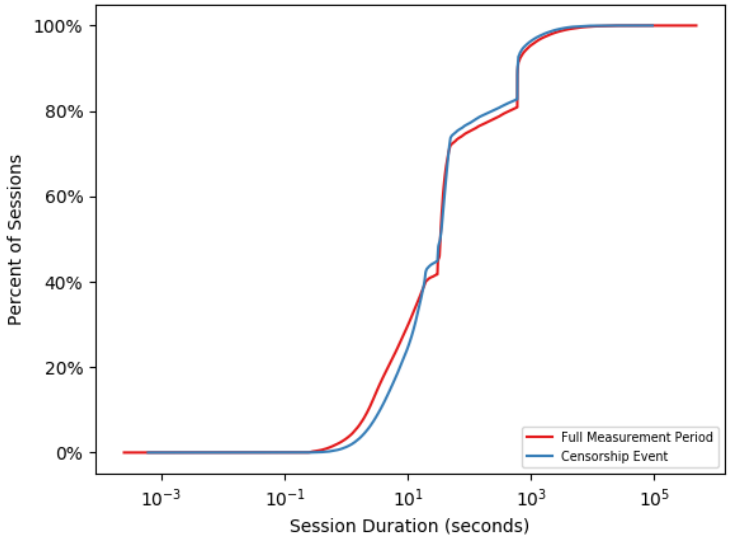


Figure 3.18: **Session Duration Distribution**— Clients see a range of connection durations over both our full measurement period and the censorship event in our last week. The median connection lasts 50 seconds. We also do not see a large effect on session duration during the increased utilization of the censorship event.

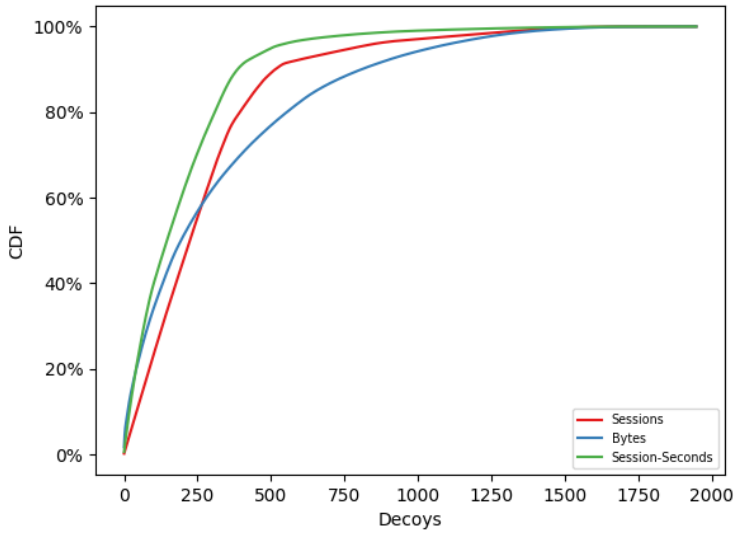


Figure 3.19: **Decoy Distribution**— We show the distribution of sessions, bytes, and session-time across the decoys in our measurement period. These lines are not as similar as those in Figure 3.14, indicating a difference in utilization among types of resource. In particular, session-time and bytes are particularly focused on the most popular decoys, as indicated by the steepness of the CDF at the beginning of the x-axis.

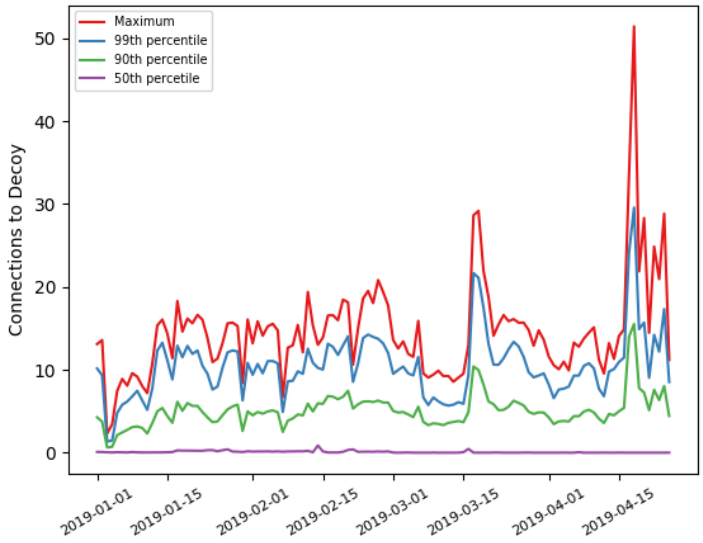


Figure 3.20: **Decoy Quantiles**— Decoy utilization is not evenly distributed across decoys. The median decoy typically has no more than two clients connected to it at any time. This does not continue to the 90th percentile.

Rank	Mean Concurrent Connections	Connections	Transfer Rate (MB/Day)
1	13.24	163,991	12.32
2	12.76	167,277	10.74
3	12.00	167,144	10.70
4	10.75	167,507	9.14
5	10.70	128,691	13.29
6	10.68	151,699	8.04
7	10.48	127,980	12.89
8	10.42	161,146	9.15
9	10.41	127,971	13.40
10	10.34	127,948	12.67

Table 3.1: **Top Decoys**— We show the 10 most frequently used decoys during our measurement period (115 Days). No single decoy takes the bulk of connections over the entire measurement period, with each of these seeing a mean concurrent connection count of over 10. Among top decoys, they are well distributed through our available address space, with the exception of decoys rank 5, 9 and 10 that are in the same /24.

We note that our mechanism for communicating and performing decoy opt-out worked; two domains included us in their `robots.txt` file and were immediately excluded from future decoy lists.

3.5.4 Station Performance

During our study, we do not find that any of the stations are saturating computation, disk, or memory resources. We confirm that the station operation itself does not impact client connectivity by having clients report the number of failed decoys before each successful connection, shown in Figure 3.22. We see that this does not change in line with our traffic volume changes. We only observe a dip when a routing configuration was temporarily used that more reliably routed decoys past our station. This change and reversion was outside of our direction. Another routing change, where a decoy prefix was announced past two stations briefly, caused a spike in the number of connections passing two stations. This tested our constellation architecture, ensuring we can handle seeing the same connection at multiple taps.

Nonetheless, the average number of failed decoys for each successful connection remains above 1, indicating that for every successful connection, a client typically tried and failed to reach our

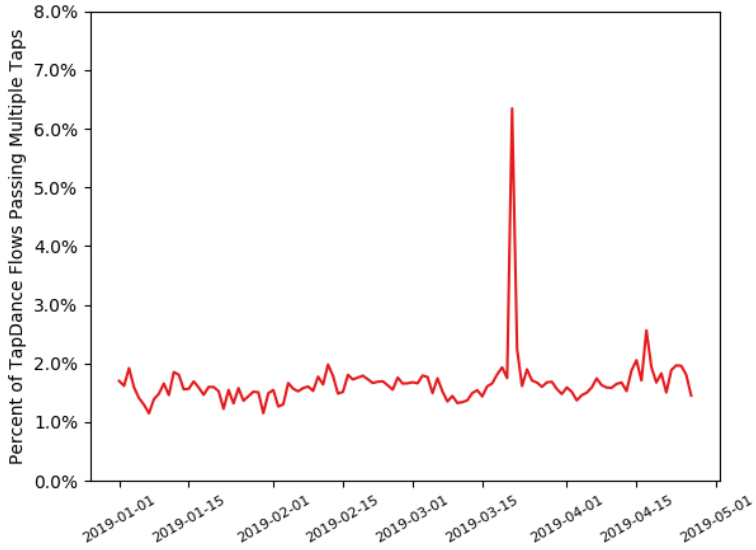


Figure 3.21: **Duplicate Detection**— Over our routing period we observe consistently low duplication of client requests to decoys, with the exception of one peak in late March corresponding to a routing change for a network containing our decoys. However, the baseline rate of 2% is high enough to justify our architectural choice of decoy station constellations.

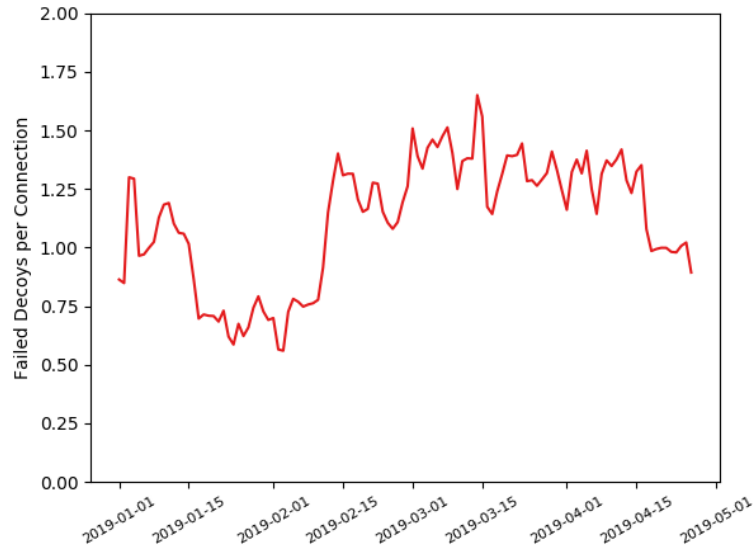


Figure 3.22: **Client Decoy Failures**— Clients select a decoy at random from a list we provide and attempt to connect. After sufficient failures, the client tries another, and repeats until it can connect. We observe that clients fail about one decoy for each successful decoy, however these are not the same decoys that fail across all clients. This indicates a significant challenge in predicting which clients can use which decoys.

proxy with another proxy. While clients currently retry until successful, this metric shows that future improvements to decoy selection would likely have a noticeable impact on latency and observability.

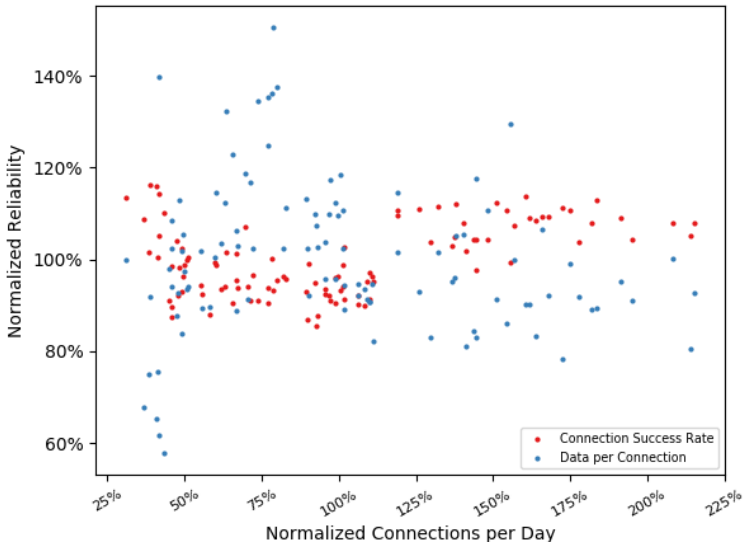


Figure 3.23: **System Reliability Under Varied Load**— We attempt to correlate our system's reliability with the traffic it handles. We quantify system reliability with the connection success rate and the amount of data per connection. All values are normalized to the mean of the measurement period. We find that the connection success rate has a Pearson's R of 0.40, with weak positive correlation of traffic volume and reliability (linear regression slope $= 0.068 \pm 0.030$). The data per connection showed a Pearson's R of -0.11 and the linear regression slope showed no significant impact.

In order to quantify the impact of the volume of traffic we serve on the quality of service, we perform correlations between the traffic volume and quality metrics. In Figure 3.23, we show a scatter plot where each dot represents a day of operation. The x-axis indicates the number of connections we handle that day, normalized by the mean value over our measurement period. The y-axis is a similarly normalized scale of reliability, as quantified by client connection success rate and the amount of data transmitted per connection. The amount of data transmitted per connection and the total connections served have a Pearson's R of -0.11. Upon linear regression, we find that the correlation effect size is insignificant ($p = 0.05$). The connection success rate and total connections served correlate with a Pearson's R of 0.41. Interestingly, this demonstrated a small positive correlation (linear regression slope $= 0.068 \pm 0.030$).

Both of these correlations support the claim that we operated within the bounds of what our system can reliably handle; neither shows statistically significant negative correlation. Qualitatively, we also observe that our plots do not have a "tipping point", after which we would see significant performance degradation.

3.5.5 Blocking Event

During the final two weeks of our study period we observed an increase in traffic. This was the result of the deployment of new censorship techniques in Iran, the country most of our users are in. While they were able to block many of the transports used by Psiphon , they did not block TapDance, so we received an increased load from users unable to access other protocols. This accounted for a 4x increase in the percent of Psiphon traffic using our system, shown in Figure 3.24

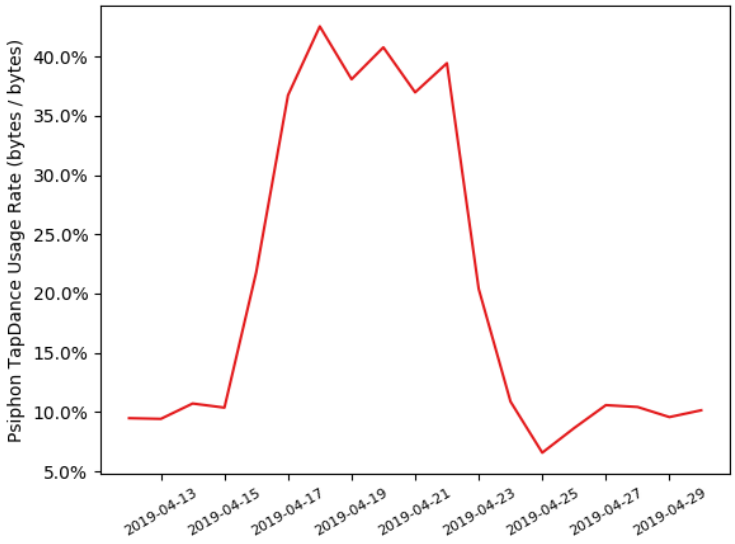


Figure 3.24: **Psiphon Tapdance Usage Rate Under Censorship**— We zoom in, with higher granularity on Tapdance usage at the end of our study period. Here, we see the censorship event and a restoration to to normal clearly.

The spike in traffic was not from a few client subnets alone. We observe similar distributions of resource usage among client subnets during the censorship period alone in Figure 3.25. We observe some client subnets maintaining longer connections, but the effect size was not enough for us to take action. Importantly, we do not see a change in byte per session distribution in any subnets,

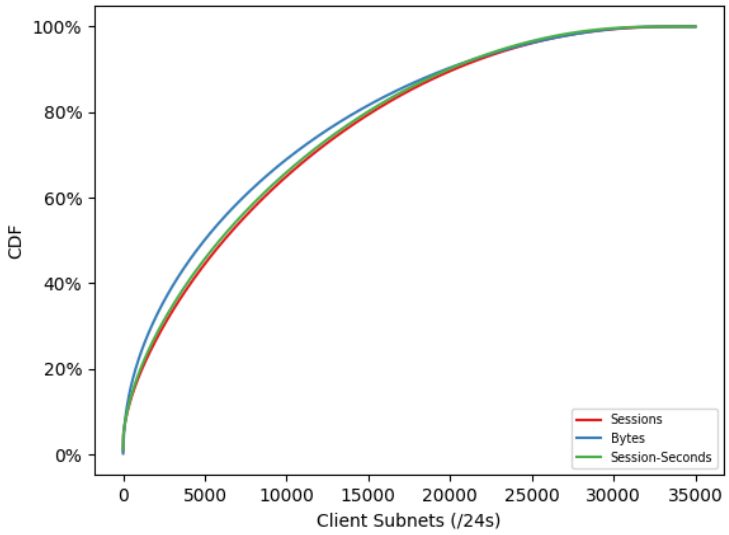


Figure 3.25: **Client Subnet Distribution Under Censorship**— We replot Figure 3.14 over just our censorship period. We see increased concentration on some client subnets, in particular among byte usage, however half of all bytes are still spread over 10% of our client subnets.

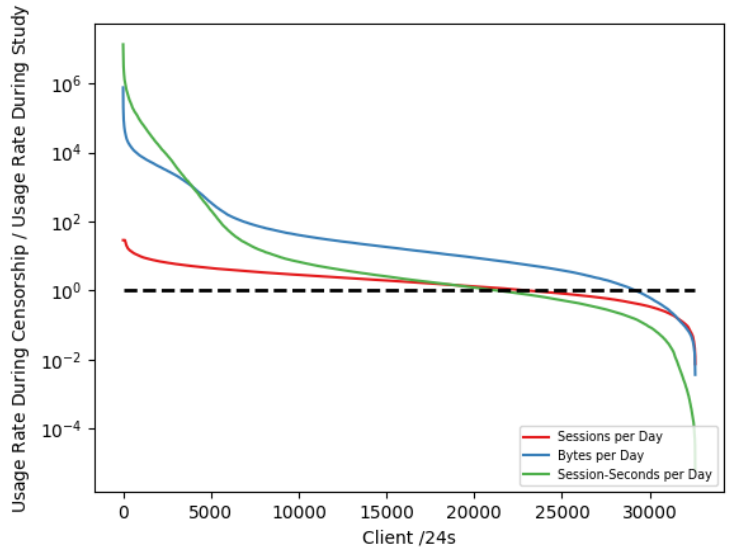


Figure 3.26: **Client Distribution Change**— Over the measurement period, most clients see an increase in sessions, bytes, and connection time. However, some clients see the opposite, decreasing in use. The clients most negatively impacted are in regions not affected by the change in censorship practice during the last week.

indicating that our change in traffic was due to an increased frequency of use, not because one subnet was using significantly more system bandwidth. In Figure 3.26, we show during censorship,

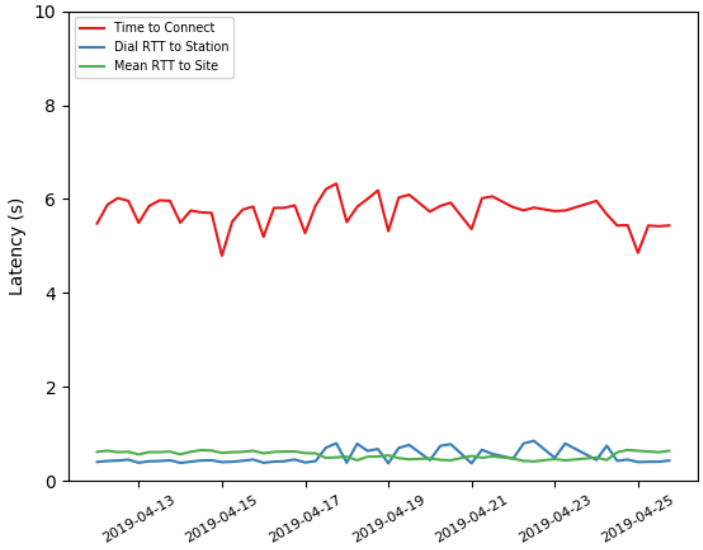


Figure 3.27: **System Latency Under Censorship**— We zoom in, with higher granularity on system latency at the end of our study period. Here, we see a slight increase in dial RTT, but it does not noticably increase time to connect.

most client subnets see an increase in use of our system. However, some saw drastic decline in their use of our system; particularly those unaffected by the change in censorship. Even the small increase in latency to dial the station under censorship load, shown in Figure 3.27, causes otherwise unaffected clients to shy away from our system to lower-latency protocols. This further suggests that efforts to decrease latency (such as better decoy selection or more aggressive timeouts for failed decoys) may increase the share of users that select TapDance instead of other transports.

3.6 Discussion

In this section we will discuss some of the lessons we learned about the unique challenges of deploying a Refraction Networking scheme in practice and what this means for the future directions Refraction Networking should take.

3.6.1 Role of Refraction Networking

During this study, Refraction Networking served a small share of Psiphon 's overall traffic. It was only deployed to a small fraction of Psiphon users and only served about ten percent of

those users' traffic over our study period. Given the time spent maintaining the deployment of this distributed system and navigating the institutional relationships to deploy it, this does not seem like a worthwhile investment on its face. However, Refraction Networking played a critical functional role during the censorship event of keeping users connected that would otherwise be censored. Keeping any line of connectivity open during censorship is vital as it allows updates to circumvention software and news relevant to the censorship event to reach the user.

3.6.2 Cost

The current deployment had a one-time-cost of about 6,000 USD per site (i.e., for Tapdance stations and central proxy). This totals 30,000 USD across our current deployment. This was used for purchasing hardware such as commodity 1U servers, network cards, optical modules, cables, etc. The total co-location cost (i.e. rack space and power) of the current deployment is estimated to be at about 13,000 USD per year. The bandwidth cost for a 2 Gbps upstream connection is estimated to be 24,000 USD per year. Finally, the ISP assigned a network engineer with an effort of about 40% FTE (on average) to help with system maintenance, management and operation.

Much of this cost is specific to Refraction schemes, such as the cost of co-location in Internet exchange points and the amount of engineer time dedication to system maintenance. While engineering costs will go down with stability and scale, the cost of operating core system architecture in IXPs incurs cost beyond other proxy schemes.

3.6.3 Partner Concerns

Our deployment partners have several very reasonable concerns when deploying refraction networking:

3.6.3.1 Will the deployment impact normal production traffic?

TapDance is specifically designed to avoid interference to an ISP's normal operation. By only observing a copy of tap traffic, TapDance station outages do not affect regular commodity traffic flowing through the ISP. However, we note that an ISP could become overloaded if enough TapDance users started using it. In our case, our ISP observed 70 Gbps of commodity traffic (of a

140 Gbps capacity), while our user traffic added only about 500 Mbps, well below a level to cause problems. We also have the ability to scale down users at a coarse granularity to mitigate potential capacity concerns, though luckily we have not encountered this issue.

3.6.3.2 How will decoy websites be affected?

Our deployment used between 1500 and 2000 decoy TLS websites, which clients connect to in order to communicate with the TapDance station. These sites do indeed see a small increase in load, but the load is spread across all decoys. We carefully monitored the number of connections to each decoy to ensure it was under a conservative threshold (see Figure 3.20). We also allow decoys to opt-out, and observed two decoys that did so, which were automatically removed from our decoy lists.

3.6.3.3 Will censors attack the ISP in retaliation?

Our ISP partner was also concerned that censors might try to attack either the ISP or the decoy websites to deter deployment. We did not observe evidence of this occurring, though we cannot say for certain censors will never respond this way. As a mitigation, the ISP formed an agreement with a DoS visibility service that agreed to support the ISP in the event of such an attack.

3.6.4 Future Directions

The lessons learned during this deployment may be useful for those looking to improve upon Refraction Networking techniques or manage further deployments. In particular:

1. Refraction Networking is useful in our deployment as a fallback technique to keep users online through heightened censorship.

2. Decoy attrition does not pose significant challenge for our scale of deployment over 18 months.

3. Router-level route prediction for client-to-decoy connections is challenging and important to the performance of our technique.

4. ISP partnerships are a bottleneck to further growth of Refraction Networking.

Of these lessons, two are major challenges we face in future growth of decoy routing: further partnerships with ISPs and locating decoys in a more generalized way using fine-grained route prediction per client. Currently our scheme only uses ISPs near the edge of the network and observes a majority of the inbound traffic so we can use the subnets they terminate as a seed for finding decoys. However, this is imperfect and still about half of the time our client picks a decoy that does not pass a station due to routing behavior. We do not currently have a reliable way to determine which network link a client will use to enter our partner network en route to a particular site, and therefore the client must guess. This is further compounded if we try to partner with Tier 1 or Tier 2 ISPs that do not have a client address space that only they serve.

Partnership with Tier 1 or Tier 2 ISPs may also raise more concern with impact to decoy websites. The lack of direct relationship to decoy websites by any decision-making stakeholder makes it less comfortable for network operators to accept potential risks on behalf of the decoys, under our opt out model.

Fortunately, both of these challenges are addressed to some extent by the recent work by Frolov et al. [80], which proposes a decoy routing scheme in which the importance of decoys backed by a real website is reduced. Rather, beyond initial registration, decoys are produced from address space with no web server present. We see promise in pursuing this scheme in deployment as a practical solution that reduces institutional concerns with active deployments. However, we note that it does not resolve the challenges we face, and there is room for improvement by future techniques.

3.7 Conclusion

Despite extensive research, Refraction Networking has had little adoption in practice to circumvent censorship. This chapter presents results from the first *continuous* deployment of a Refraction Networking scheme to users *in situ*, in operation for over 18 months. Our experience demonstrates that Refraction has played a successful role to provide connectivity, even when censors increase their activity, and helps inform which aspects of new techniques will help ease of further deployment.

CHAPTER IV

Measurement of Web Privacy Defenses

While there are competing definitions of privacy in the literature, nearly all of them are violated by the online tracking ecosystem. Users are unwittingly being tracked nearly everywhere they visit online by companies they may not know the name of, let alone have agreed to let them build profiles of them for advertising purposes. These trackers even take measures to deliberately circumvent user choices to avoid being tracked. There is significant information asymmetry between these trackers and the typical user online.

Our work's goal is to reduce that information gap in at least one way: learning which browser extension is most effective at hiding a user from advertising and analytics companies. We quantify the extent to which ad-blockers, AdGuard and AdBlock Plus, provide very little privacy despite advertising privacy as a benefit of their product. Further, we refute the presumption of prior work that this behavior is due to "Acceptable Ads" programs with a more accurate model of these blocking extensions and improved scale. Experts now have data to back up the recommendation of uBlock Origin and Disconnect.me for user privacy. This also justifies Firefox's choice to adopt the Disconnect.me list as a default blocking behavior. In spite of its narrow focus, it is one of the strongest tools at our disposal.

This chapter is adapted from a joint publication currently under submission [201].

4.1 Introduction

Advertising is an important piece of the current web experience and economy. Most users' search engine is Google and its parent corporation, Alphabet, drives its 800 billion USD market cap by deriving 83% of its revenue from advertising [10]. Users have a complicated relationship with ads, and for good reason: despite being intrusive on their experience, ads can provide benefit if they are relevant [194]. It is in search of relevance by advertisers for being profitable, that many online advertisers target ads. However, underlying the targeted advertisements are user profiles built to enable the targeted ads.

The underlying profile building and tracking that supports advertisements is where the users can come to harm. Anecdotally, pregnancy status [95] and sexual orientation [93] have been disclosed accidentally without direct access to the user's profile. In one study, a majority of users were uncomfortable with any targeted advertising based on common interests inferred by advertisers. These same users were made even more uncomfortable by interests pertaining to their health or religion [52]. This says nothing of inferences that combine credit card history, voter registration, and browsing behavior used by journalists specifically to demonstrate the power of targeted advertising [188].

In lieu of a strong model of consent, some users have turned to using browser extensions to prevent being tracked. This is not to be confused with the separate but related phenomenon of ad blockers [137]. It is easy to confuse the categories of blocking extensions: ad blockers advertise their utility in preserving privacy [5, 72] and use some of the same rules to block requests.

Researchers have studied several aspects of these browser extensions and the block lists that direct them how to behave. This even includes some comparative evaluation of these extensions to inform recommendations to users. However, these comparisons do not do enough to look into the factors that contribute to the extensions' effectiveness, leaving us asking the central question of our work: what are the roles of manual list creation, organizational mission, and the connections explicitly white-listed from blocking in the limits of ad blocking extensions as deployed today?

In this chapter we look to understand blocking extensions better and unpack these factors as they

impact extension efficacy. To do this we visit a list of the top million sites with Firefox. Our model of the blocking extensions behavior is more accurate than prior work. We also compare browser extensions' blocking behavior over all rules defined in each of their rule-sets, reducing reliance on crawls limited in scope and subject to transient advertiser practices.

We infer how many page visits would be disclosed to tracking companies over our crawl, and use this information to evaluate the browser extension efficacy. Browser extensions are evaluated both with and without their Acceptable Ads programs enabled, and find results that contradict those of prior work: Acceptable Ads programs are not the primary reason for their extensions' poor performance.

We use the inclusion graph of our crawl to identify advertisers not in the hand-curated lists used by blocking extensions. The size of these newly discovered advertiser lists help us understand the completeness of the manual curation of the lists over their target communities. When combined with our understanding of Acceptable Ads programs and the stated missions of the browser extension authors, this lets us better understand the factors that contribute to extension efficacy.

Our work answers three questions that had not been previously answered. *1)* How effectively do the full population of A&A companies track users across the web? *2)* What tools are most effective in stopping this tracking? *3)* What factors of the effective tools most contribute?

4.2 Related Work

In this section we describe the research that comprises the background of and is related to our own work. First, we describe the initial work in web tracking. Second, we discuss several related subfields that have emerged in studying web tracking. Third, we discuss the current state of research in evaluation of blocking extensions to prevent web tracking. Finally, we discuss the research that we most directly draw from.

Much work has been done analyzing the technologies used to identify users across the web. The academic study began in 2006 with a mention of the potential of tracking users between page visits by Jackson et al. [109]. Work focused entirely on tracking as a practice began soon after with the

work of Krishnamurthy et al. [122] and Jensen et al. [111]. Krishnamurthy and Willis then outlined concerning trends of centralization and the weak state of defenses at the time [123]. Research soon formalized issues in measurement methodology [91], identified browser fingerprinting as a tool in the arsenal of trackers [62], identified avenues of instrumentation of JavaScript to track information flows [110], and asked questions of user perceptions of the early targeted ad ecosystem [140] setting the foundations of the following decade of work.

One subarea of web tracking research is the human interaction with web trackers and defenses. Some work has focused on user understanding and perceptions of targeted ads [39, 142, 194, 213], finding significant gaps in knowledge and inconsistent user mental models. Further work focused on sensitivity to various targeting techniques [7, 128, 133], discovering areas users were particularly unwilling to have be used in targeted advertisement, such as traffic from dating sites or length of retention. More recent work [52] focused on similar questions, but focused the line of questioning on the underlying inferences used to build profiles. More closely related to our work, research has focused on user adoption of both ad blockers [168] and blocking extensions more generally [137] finding diverse reasons for adopting these extensions and little issue with website breakage. Finally, the justification for inclusion in the Acceptable Ads program by AdBlock Plus was refuted [207].

Some studies have focused on cookies, one of the most conspicuous means of identifying users uniquely. Much of the research has looked at identifying unique identifiers in cookies and how third parties share these identifiers to synchronize identities across multiple tracking networks [31, 74, 84, 158, 174, 221]. Some work in this area has focused on policy, specifically considering implications of cookies on governmental mass surveillance [68] and EU privacy law [47, 192, 195].

Other work identifies ways to uniquely identify users other than cookies, broadly termed browser fingerprinting [2, 29, 153, 214]. Further research showed how these techniques changed over the history of the Web using the Wayback Machine archive [106, 129]. Measurements scaled significantly in this area [67, 178], giving us the tool we use for our measurements, OpenWPM [67]. Additionally, some research noted the use of other state than cookie stores to create persistent

tracking mechanisms [1, 139].

Much research has been performed to understand the behaviors, weaknesses, and extent of the advertising ecosystem [20, 22, 32–34, 85, 126, 159, 170, 204, 215]. In particular, some work focused on out-of-band exchanges of information between advertisers [21] and challenges in attributing advertisements [119]. These show limitations in fully automated, but much larger measurement techniques such as our own. Also, some research has analyzed these same questions for defensive countermeasures used by ad networks to circumvent blockers [107].

Several new solutions have been proposed to defend against web tracking in the research literature. Several new ad network systems have been proposed that cryptographically remove traffic disclosure while continuing to allow targeted advertisement [17, 77, 92, 190]. Others used machine learning to identify the advertising requests to be blocked [105, 108, 115, 131, 147]. Nikiforakis et al. proposed entropy injection into the features used to fingerprint users [152], however further work showed how to detect the entropy and retrieve the original values [202]. Recent work by Zhu et al. has shown promise in stealthy blocking techniques [219], circumventing defensive countermeasures used by ad networks to circumvent blockers.

We draw much inspiration from the work of Bashir et al. [23]. In particular, the view of the entire advertising ecosystem as a graph, similar to that of Kalavri et al. [115] and Gomer et al. [86], shaped our thinking. Kalavari et al., however, focused on automatic identification of trackers using machine learning to replace current blocking extensions, supporting this by analyzing the referrer graph structure. Gomer et al. studied a different graph representation of the tracking ecosystem, creating a bipartite graph of first-parties and the third-parties they load, rather than the structure of the third-party ecosystem itself. Bashir et al. study the same graph that we do and compare some blocking extensions. However, by focusing our study on the blocking extensions and by studying a different, more diverse, and larger population of web pages with a more accurate blocking extension model we were able to come to deeper conclusions about blocking extensions and refute proposed explanations for findings of that work.

The closest research to our own is in the evaluation of existing solutions to online tracking.

Early work identified the futility of the DNT header [19]. Other studies focus on the interaction of ad-blocking tools and advertisers in an arms race of detection [143, 154]. However, these fail to account for out-of-band communication between A&A companies and did not explore causes of ineffective blocking as deeply as in this work, also identifying Acceptable Ads as the reason for AdBlock Plus's poor performance. The work of Malloy et al. [134] focuses on the market penetration of ad blockers, and focuses on impact of advertisements displayed, not tracking. The work of Karaj et al. [117] and the work of Yu et al. [215] focus on the users of a single browser extension which, while very useful, is biased in uncharacterized ways and did not try to generalize beyond their primary study population. Gervais et al. [83] and Bashir et al. [23] perform less focused studies over a very small number of popular websites. We demonstrate in this work that our population of trackers discovered is larger than that measured in these papers and that tracking is quantitatively different from the top thousand to the top million (which are equally weighted under typical Zipf-ian assumptions). Merzdovnik et al. [143] provides a focused analyses of a large-scale top domain list. However, they do not study the A&A ecosystem outside of 30 actors and additionally fail to analyze how information is shared in an RTB system. This work and that of Bashir et al. [23] show that out-of-band RTB sharing has significant impact on user privacy. However, they do not explore deeply enough which tools are more effective at stopping tracking and why, only analyzing one ad blocker and one privacy tool, and also concluding incorrectly that Acceptable Ads are the reason for AdBlock Plus's poor performance. Bashir et al. [23] also used a less accurate representation of the acceptable ads list and a data set focused on popular online shopping sites, which led to a limited scope of A&A websites. Walls et al. [207] studied the impact of acceptable advertisement lists, but focused on the composition, evolution, and user impressions of the list; this omits analysis of the impact of the list on user privacy. Vastel et al. [203] studies filter list composition, but pursues questions on how to trim filter lists of rarely used rules. This did not consider the impacts to user privacy due to out-of-band communication by A&A companies or the most effective trackers and did not seek to explain why different tools were more effective. This insights of our work are complementary to those of Vastel et al. for these reasons.

4.3 Methods

To support the analyses in the remainder of this chapter we employ methods we develop ourselves as well as methods taken from prior work. We present the methods in this section, referencing them in later sections as they are used.

Throughout we use the acronyms FQDN and eFLD. These are the Fully Qualified Domain Name (e.g. `www.foo.co.uk`) and the effective First Level Domain (e.g. `foo.co.uk`).

4.3.1 Web Crawl

Prior work has studied many websites' tracking behavior using OpenWPM, an instrumented browser measurement platform [67]. We re-use this tool and its standard data formats, allowing re-use of our analyses over future and past crawls. To facilitate this, we release our source code and data at [redacted for anonymous submission].

Also like prior studies, we crawl the homepages of a large top domain list. We use the Tranco list [1] [125] created on 07 July 2019. This improves reproducibility while maintaining the scale of the largest prior work and reducing bias from domain fluctuations known to be in the more commonly used Alexa top list [9, 177]. We compare the top crawl targets and advertising and analytic domains contacted in our study to the most recent publicly available Alexa crawl [67] in Table 4.1.

We attempt to load the HTTP version of all one million domains with OpenWPM in a stateless crawl using Firefox, storing HTTP request, response, and redirect data. We performed these measurements from a dedicated research scanning subnet from 12 July—27 July 2019. Of the 1M input list, OpenWPM received responses for 895,856 sites, missing 10.5% due to timeouts, unavailable destinations, and various other errors; this error rate is similar to prior measurements of top million sites. This results in 89,136,479 total requests during the crawl to 62,970,931 unique URLs.

[1] Available at https://tranco-list.eu/list/J96Y.

	Alexa (Jan 2015)		Tranco (Jul 2019)	
	Targets	Trackers	Targets	Trackers
1	google.com	google-analytics.com	google.com	doubleclick.net
2	facebook.com	doubleclick.net	netflix.com	google.com
3	youtube.com	facebook.com	facebook.com	facebook.com
4	baidu.com	googlesyndication.com	youtube.com	googlesyndication.com
5	yahoo.com	googleadservices.com	twitter.com	google-analytics.com
6	amazon.com	google.com	microsoft.com	youtube.com
7	wikipedia.com	twitter.com	wikipedia.org	facebook.net
8	qq.com	adnxs.com	baidu.com	pubmatic.com
9	google.co.in	blekai.com	linkedin.com	adnxs.com
10	twitter.com	mathtag.com	instagram.com	wp.com

Table 4.1: **Comparison of Top Domain List to Prior Work**— We compare both the top ten in each list, accounting for 20% of Zipf-weighted visits, and the top ten eFLDs blocked by EasyList and EasyPrivacy as trackers.

4.3.2 Blocker List Testing

We compare six different blocking extensions, selected for their popularity: AdBlock Plus [72], AdGuard [5], Disconnect.me [49] (whose rules are used by Firefox in its tracking protection [145]), DuckDuckGo [54], Ghostery [42], and uBlock Origin [97]. We evaluate each tools' default configuration; however it is possible for users to improve the efficacy of each extension and even import filter lists from other extensions in most cases. We also use EasyList and EasyPrivacy [187] throughout our evaluation to identify previously known A&A domains. EasyList and EasyPrivacy are community created lists for blocking advertisements and tracking respectively, and are imported by some tools to verify We define *previously known A&A domains* heuristically as in prior work [23]: any domain that is blocked more than 10% of the time by EasyList and EasyPrivacy. We use blocking lists retrieved on April 23rd, 2019 for all extentions.

Our method of evaluating blocking lists is an improvement over prior work. We more closely reflect the actual behavior of the blocking extensions while still testing over 29M requests. We enabled this in two pre-computation stages: normalizing each tool's behavior to a single syntax and reducing sets of regular expressions into single prefix-structured regular expressions.

4.3.2.1 Normalizing Blocker List Syntax

Before we take any steps to speed up our request testing process, we first standardize our blocking list syntax. Fortunately, three extensions (AdBlock Plus [73], AdGuard [3], and uBlock Origin [96]) of the six we study and EasyList and EasyPrivacy use a very similar syntax. While these syntaxes vary, the commonly used features are shared.

The syntax for blocking rules we normalize to is a shared subset of these powerful syntaxes. Each rule is a newline-delimited string with two parts, separated by a $ character. The first part is a simplified regular expression syntax and the second part is a series of flags and corresponding values. The regular expression syntax has only five special strings that are not interpreted literally: "`||`", "`|`", "`*`", "`^`", and "`@@`". Each of these have the same meaning among all of the tools and either have a direct translation to standard regular expression syntax or indicate that the rule should be inverted, specifying requests to allow rather than to block ("`@@`").

The flag portion of these powerful syntaxes is often overlooked; it is not mentioned at all in the application of rules in prior work. In fact, the syntax for AdBlock Plus, AdGuard, and uBlock Origin vary primarily in what flags they support. We support the following flags: `important`, `domain`, and `third-party`. Respectively, these rules bypass whitelisting caused by other rules, enforce the rule only on requests originating from specific domains, and enforce the rule only on third-party requests. The only rules we ignore that could be used to aid privacy and are used in more than 1% of rules are those that block requests for types of resources, i.e. `script`, `image`, etc.. This leads to over-estimation of the power of the blocking tools, however the tools are treated equally.

Our implemented subset of these rules also elides an entire class of specification: cosmetic rules. These rules mark elements on the page as `hidden` and do not prevent requests from being sent. Therefore, we find it acceptable to omit them entirely for measurement of web tracking.

Tools that do not use this standard syntax are converted into the standard syntax. We take each of the specified rules and convert them into an identical rule that blocks exactly the same resources and subdomains, using only the limited regex syntax and `domain` flag. These rules can then be treated as the other canonicalized rule lists are in the subsequent steps.

4.3.2.2 Efficiently Evaluating Regular Expressions

During initial experiments it became clear that evaluation of all rules over all requests of our crawl performed naively would be impractical. Evaluating 89M requests against over 130K regular expressions proved to be a significant bottleneck, even with early-exit optimizations. To speed up our evaluation, we reduce the number of comparisons required in a few ways that are layered hierarchically.

First we note that some rules are only in use on very few domains. To take advantage of this, for each tool's rule list we create rule lists for each domain explicitly mentioned in the constraints. Those rules we do not have any constraint on, or only whitelist explicit domains from having the rule applied, are added to the default collection which is applied to all domains. In effect, we have created a blocking list for each referring origin that would change the behavior of the blocking tool. These lists being much smaller allows much less comparison in an initial check that may return a result quickly. If a result is not obtained from the first check, we then compare on the larger list that contains rules that apply to all referring origins.

Next we note that there is a priority order of rules that goes from a small subset to a large subset as the priority decreases: another opportunity for early exit. First, we can search in the set of "`important`" rules that bypass whitelisting, and are the smallest set. Then we can look at the whitelist rules, those that begin with "`@@`". Finally we compare the remaining rules. To allow this ordered evaluation, at this stage we further partition the rules along these lines, within each domain constraint's list.

Finally, and perhaps most importantly for our speed, we reduce all of the regular expressions in each partition into a single regular expression. Doing this naively, simply listing each expression, and separating them by '`|`' in standard regex syntax, would provide little speedup over comparing each rule to the URL. This is the exact underlying behavior the regex engine takes in its evaluation. Instead, we convert the rules into a prefix tree of their equivalent regular expressions, then convert this prefix tree into a reduced regular expression. For example, rather than converting a search for either "food" or "fold" into "`(food|fold)`", this is converted into the regex "`fo(ld|od)`",

removing duplicated comparison to the common prefix “fo’. This takes ideas from prior work in optimizing regular expression validation [18], without having to write a custom engine.

So, in review, to compare a single URL requested from a given origin, we take the following steps: first find the longest FQDN with domain constraints for which our referring origin is a subdomain and check if it is blocked or explicitly whitelisted by that structure. If not, or if there is no such subdomain, check if it is blocked or explicitly whitelisted by the rules with no domain constraints. Both of these checks are done by evaluating the next layer of the hierarchy: comparing against the important rules, whitelist rules, and other rules of that structure in that order. Again, to do this we proceed down the structure, comparing against the regular expressions built for this constraint and priority, and returning whether or not a match was found. Finally, if the result is that we found no matching rules, we interpret this as an unblocked request. In this way, we have traversed only the relevant partitions of the rules we created for this blocking tool.

While this is more complicated, it does capture the entirety of our normalized subset of the blocking syntax. Additionally, it sped our evaluation of requests from our crawl by *over 1000x*, letting us finish evaluation of all six tools in under six hours on a single 8-core Intel Xeon E5–2690. This made the comparison of multiple tools over a very large crawl practical.

4.3.3 Blocker List Probing

Initial analysis of how each tool performed on our crawl revealed that most rules blocking entire domains were not used at all during our crawl. We want to present an alternative perspective when comparing tools that is not biased by the particular crawl. To do this we created probing lists from each tool’s blocking rules.

Each blocking rule that includes a specified domain name is mechanically converted to a set of rules that capture some of the cases a rule may apply to. Multiple transformation may occur for each special character, causing exponential growth in the number of special characters. In practice each probe list does not grow impractically large.

Rules that begin with the special character “`|`” simply have the character removed. Rules that begin with the special sequence “`||`” have the sequence replaced by both “`http://`” and “`https://`”.

The special character "^" is replaced by a "/" and optionally removed if at the end of the rule. Finally, the "*" is replaced by an empty string, "1", "/", and the random string "b75de6c2/2ce0bae5" which occurs nowhere in the block list definitions. These simple transformations capture much of the diversity of behavior of the limited syntax provided by the blocking extensions. For example, the rule "||foo.com^" is expanded to the following probing links: "http://foo.com/", "http://foo.com", "https://foo.com/", "https://foo.com".

Rules are tested with referring origins taken from the domain restrictions and an unrestricted referring origin. We group the probes by eFLD of the request and compare two test lists by checking if all test probes in the same eFLD have the same outcome.

4.3.4 Inclusion Graph

Prior work has demonstrated use of the inclusion graph to demonstrate interrelationship among known A&A domains [23]. The inclusion graph is a representation of a web crawl where parties (in prior work canonicalized domain name) are node and directed edges indicate that the source party initiated a request to the destination party in some page visit of that crawl. This is in contrast to the referrer graph where an edge indicates the source appeared as the referrer in the destinations' request. JavaScript requests are a source of disagreement between these two graphs and are used often in the A&A ecosystem. We construct the inclusion graph similarly, however we our graph is over all third-parties, rather than only known advertisers. We are also able to analyze the subgraph of only known advertisers for direct comparison.

The graph we produce represents only the sample of domains we perform; there may be some number of nodes or edges not included in our sample that exist in the population. We can tell if the number of nodes and edges continue to increase as we near the end of our crawl. We show the growth of these through our crawl in Figure 4.1.

Our crawl does not saturate the number of eFLDs nor edges between them from our population. This is despite finding a larger number of them than prior work that found saturation in its inclusion graph. This is because we study a much larger population that is not restricted by type of domain or to only the top few thousand domains in a rank list.

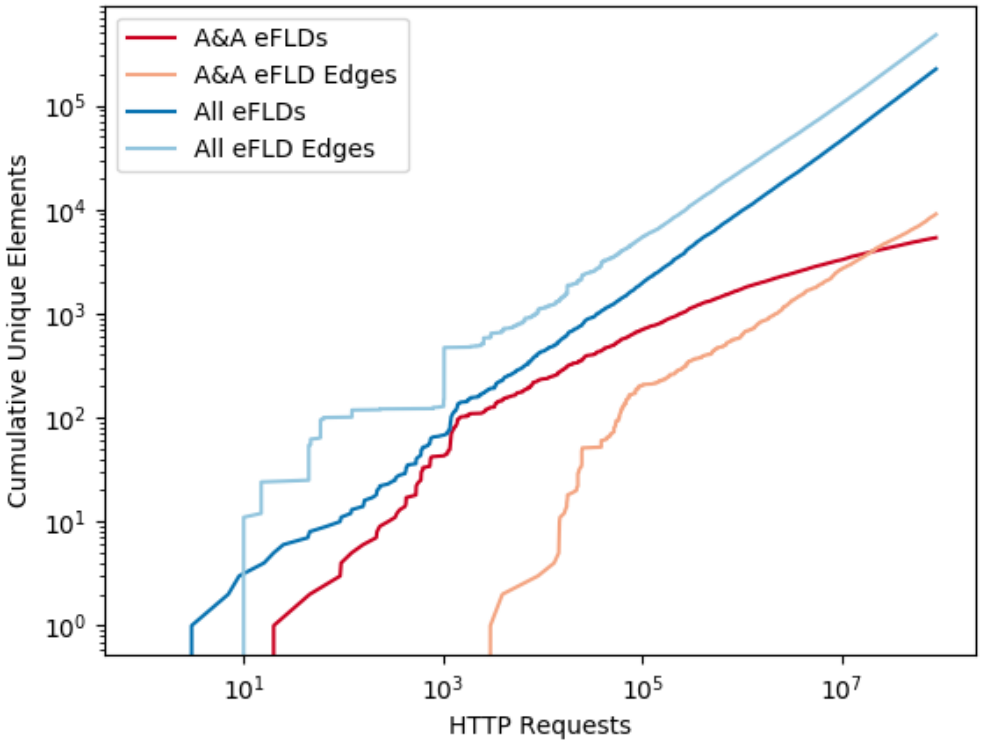


Figure 4.1: **Cumulative Unique Inclusion Graph Nodes and Edges**— We graph the cumulative number of unique eFLDs (dark lines) and inclusions between eFLDs (light lines) as we conduct our crawl. Blue lines represent the graph of all third-parties and red lines represent the subgraph of only third-parties that are known A&A domains.

This does not preclude use of the graph for the uses in this chapter, however we do not compute graph statistics to compare to prior work as they would not be representative.

4.3.5 Privacy Metrics

Finally, we analyze the impact of blocking strategies on our crawl. Prior work has defined disclosure of a page visit, sometimes termed impression, and we use these definitions. In particular, an A&A party learns that a particular user has requested the main page of a particular visit if in the course of that visit it is sent a request. Additionally, an A&A party may learn of a request if it is shared with them through other means. We consider two such means in this work[2]: **Cookie**

[2]Both of these are taken from Bashir et al. [23], corresponding to Cookie Matching and Real-Time Bidding – Constrained respectively.

Matching and **Real-Time Bidding**.

In the **Cookie Matching** model, we assume information sharing along all directed edges known to perform ad-redirection [21].

In the **Real-Time Bidding** model, we attempt to capture information disclosure that occurs when ad exchanges solicit bids from Demand Side Platforms and advertisers for a given impression. To do this, we designate some eFLDs as belonging to ad exchanges, using heuristics described in Section 4.5, and have these exchanges disclose all visits they are aware of to all eFLDs they include during our crawl. We investigate the sensitivity to the number of ad exchanges included in Section 4.6.1.

The metric we use to evaluate any A&A domain in our crawl is how many page visits cause the domain to receive a request or be further notified by another third party. To measure if a domain would receive a request, we build the inclusion graph for just that page visit and compute the reachable origins if only non-blocking requests and out-of-band disclosures proceed.

Performing this inclusion graph construction and evaluation of sharing is also expensive to do at scale. To cope with this we only perform these computations over a random sample of $54,000$ visits, weighted with a Zipf-ian distribution over their top site rank to approximate user traffic patterns. We graph several cumulative distributions of this metric over previously known A&A domains. By the Dvoretzky–Kiefer–Wolfowitz inequality this places the entire CDF within the line width on our figures with 95% likelihood, even with Holm-Bonferroni correction. As such we present the distributions without explicit error bars.

4.4 Comparing Blocker Lists

In this section we lay the foundation of our new understanding of blocking extensions by comparing the blocking lists that define the behavior of these extensions. We evaluate the extensions' behavior both over our web crawl and our constructed probing lists.

4.4.1 Default Extension Behavior

Blocking extensions rule lists have been compared to each other in prior work in their rate of blocking and the privacy that affords. However, some aspects of our comparison are novel. Our comparisons are used to help disambiguate the impacts of different factors on the privacy provided by these tools. Also, our evaluation uses rule lists to generate test lists providing a larger perspective and one that lets us ignore issues of the coverage of a single crawl. Finally, our data set enables insight into a larger test list than prior work focused on blocking extensions, and the first top-million study of web tracking in four years.

An important distinction must be made when evaluating the size of lists is whether the measure used is one that is evaluating the list itself or is grounded in a sample of requests. We compare list sizes in Table 4.2 using both.

The largest lists by size and by domain explicitly mentioned in their rule set are AdBlock Plus and AdGuard, however it is not the case that these most active filtering tools based on our crawl. AdGuard blocks the second fewest requests to a relatively diverse set of domains while one of the smallest lists, Disconnect.me, provides the most requests blocked. This disagreement highlights a difference in strategy between growing a large test list that does not block entire domains as third parties and entirely blocking a few major actors.

While comparing the list sizes by measure does tell us some differences between the tools, it does not show the intersections of each tools' behavior. To do this, we must compare this tools pairwise. Understanding the relative intersections of tools exposes which tools may source their rules from one another and how independent other tools are in their behavior.

	AdBlock Plus	AdGuard	Disconnect.me	DuckDuckGo	Ghostery	uBlock Origin
Canonicalized Rules	50,844	51,552	2,568	1,789	4,667	65,772
Explicit eFLDs	28,529	31,427	2,117	1,736	3,658	37,912
Blocked Requests	9,580,925	12,276,925	30,840,615	20,073,773	24,019,804	25,692,118
A&A eFLDs	2,545	2,981	1,340	943	2,190	6,115

Table 4.2: **Tool Block List Sizes**— We compare six browser extensions' block list sizes and coverage in our crawl data set. Our counts are by rules after canonicalization, explicitly mentioned eFLDs in rules, requests blocked, and the number of eFLDs with at least 10% of requests blocked.

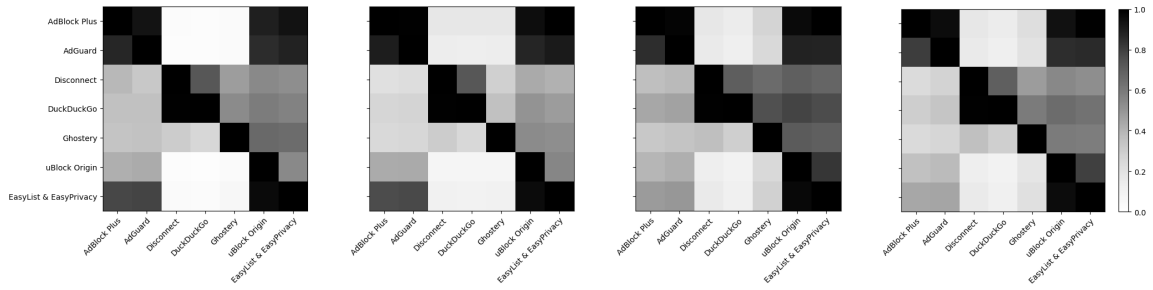
There are subtleties in comparing different tools under metrics that can be described other than "blocked" or "not blocked". For example, comparing the domains that two tools block can be done by taking the domains mentioned explicitly in each tools' test list, however test lists can hold exceptions for many domains and may have more generic rules that do not explicitly mention all of the same domains. To counter this, we can compare the domains that are blocked under at least the same conditions as the other test list. This can be done using our probing lists described in Section 4.3.2. Note that this comparison is asymmetric, e.g. only Tool 1 may block `example.com/foo` and only Tool 2 may block `example.com/bar`.

In Figure 4.2a we compare tools by which domains where tools' test lists explicitly block any resource from a given domain and in Figure 4.2b we compare the relative coverage of those lists using our generated probe sets from Section 4.3.3. We include EasyList and EasyPrivacy as a baseline as it is used in prior work as a baseline.

In both figures the vertical bar corresponding to uBlock Origin's performance on their peers' test lists show the most consistent coverage. It has the strongest definitions when evaluated on other tools' probe lists, with the exception of Disconnect and DuckDuckGo's similar definitions. uBlock Origin shows similar behavior to AdGuard and AdBlock Plus: all three appear to be derived from EasyList with varying degrees of difference. Notably, uBlock Origin is the only that gets stronger when diverging from the EasyList definition, observed by comparing squares reflected over the diagonal line that is all black.

Additionally, we observe that Disconnect and DuckDuckGo have similar list definitions, with Disconnect's definitions being stronger. It is not explicitly stated that one derives from the other, however this is one simple explanation.

While these comparisons describe the broadest possible coverage of each list, we are not able to infer how these lists compare on resources from real websites. To do this, we must utilize our crawl data set. We perform similar comparisons with this practical data: Figure 4.2c shows how many domains have at least one domain blocked by both tools and Figure 4.2d shows the number of domains that are blocked in at least the same cases for a pair of tools. Both values are taken as a



(a) Definition - Any (b) Definition - Covered (c) Measured - Any (d) Measured - Covered

Figure 4.2: **Comparing Extensions Pairwise**— We compare each extension to all others, and EasyList and EasyPrivacy in four different ways. The comparisons are made over both the definition of the rules and how they behave in our crawl (Definition vs Measured) as well as if the tool blocks the domain at all versus if the tool blocks the domain in at least the same instances as the tool it is compared to (Any vs Covered). For example, the entirely black square in the upper right corner of *(c)* means that for every eFLD we observed a request blocked by AdBlock Plus, we also observed a request to that same eFLD blocked by EasyList and EasyPrivacy.

fraction of the number of domains blocked ever by the baseline tool.

In our crawl, the strongest tool, uBlock Origin, performs even better, as does Ghostery. Overall our observations from the comparison of list definitions are borne out in our crawl data.

The smaller test lists by rule count perform poorly against the domains blocked by larger test lists in these figures. We count by eFLDs, so this does not reflect total requests blocked, nor does it reflect the privacy impact of blocking each domain. In Section 4.6, we will analyze the privacy benefits of these tools.

4.4.2 Explicit A&A Exceptions

Some blocking extensions are straightforward in their effort to reclaim user experience and privacy without consideration for the advertising and analytics companies. Others explicitly exempt some requests as acceptable, despite their known analytic or tracking nature. While these exceptions are made for a variety of reasons, we compare them directly to determine their impact. To clarify, we do not include all rules that white-list requests; their exclusion may be because they are not related to tracking despite being caught in another rule, to circumvent anti-ad-blocker tools, or to allow an analytics company to perform analytics on its own domains. These are all examples of

what is not exempt A&A behavior.

In order to understand the impacts of the acceptable advertisement lists, we repeat some of our analysis in the previous section, removing the explicit exceptions. Only AdBlock Plus [71], AdGuard [4], and Disconnect [48] claim to have these explicit exceptions.

We show differences in list size in measures affected by the removal of explicit exceptions. Table 4.3 shows that only AdBlock Plus is affected significantly by their "Acceptable Advertising" list. AdGuard showed only 30 rules that reference 10 domains removed, and none were utilized during our crawl.

Disconnect.me showed effectively no change in its behavior with and without its exception list removed. The default behavior of Disconnect.me's browser extension and the content of its publicly available block list include those sites it claims it does not block in its list. This means that when we introduce the domains claimed to be excluded from their default list, we gain 4 new rules and 508 duplicate rules. The number of rules went up as an artifact of how we constructed the test list; our default test list includes only those that ship with the browser extension, with no care taken to exclude the alleged exceptions. This resulted in 3 new domains being blocked in our crawl and a 0.2% increase in blocked requests.

If AdBlock Plus were to block its "Acceptable Advertising" it would increase the requests blocked by 20% and while only increasing the domains that have 10% of requests blocked by 1%. These eFLDs are all blocked in over half of requests by EasyList and EasyPrivacy. Included in these are eFLDs with more than 50,000 requests: `yahoo.com`, `media.net`, `linkedin.com`, and

	AdBlock Plus	AdGuard	Disconnect.me
Canonicalized Rules	-8,468	-30	+512
Explicit Domains	-1,892	-10	+4
Blocked Requests	+2,365,442	0	+55,248
A&A Domains	+26	0	+3

Table 4.3: **Tool Block List Size, Explicit Exception Impact**— We show the changes to the scope of each blocking tool when their explicit exceptions are removed. Counting is performed identically to that of Table 4.2.

bttrack.com.

We do not find agreement with prior work on the scope of the Acceptable Advertising list in our more precise model. Prior work [23] included major ad networks in the domains excluded by Acceptable Advertising, however we find that this is not the case in practice. This is due to our more faithful representation of the block lists, in particular implementing domain restrictions; the ad networks do show up in our Acceptable Advertising list, however the vast majority are restricted to being whitelisted on a small number of websites. Two notable exceptions are Google AdSense[3] and subresources of some advertisers used to monitor advertisement clicks[4], which are permitted on all but a few sites. Despite these broad exceptions and their potential for misuse by advertisers, ad networks are still blocked by AdBlock Plus because of the broad restrictions on most exceptions. This refutes a previous belief that AdBlock Plus performs poorly *because* of the Acceptable Ads program.

4.4.3 EasyList Variants

Three tools we test are at least in part based upon EasyList or EasyPrivacy: uBlock Origin, AdBlock Plus, and AdGuard. Notably, uBlock Origin is the only of the three that adds blocking behavior to EasyList and EasyPrivacy combined. It adds separate resources for malware risks, privacy, and resource abuse above its bulk import of EasyList and EasyPrivacy.

Both ad blockers are based upon EasyList, but do not import EasyPrivacy. However, EasyList blocks several ad exchanges that are allowed by the ad blockers. AdBlock Plus directly imports EasyList, then adds exception rules for some ad exchanges and websites. AdGuard, on the other hand, imports EasyList, then applies changes to form a single modified list. Their behavior blocking ad exchanges compared to EasyList and EasyPrivacy is shown in Table 4.4.

Some ad exchanges, like adnxs.com, show no change in behavior in ad blockers while others, like yandex.ru, show complete whitelisting. More interesting are those that show partial reduction of blocking. We observe through manual inspection that the vast majority of the partial reductions

[3]This change was not discussed in a public forum: https://hg.adblockplus.org/exceptionrules/rev/c182d9dc5600.

[4]This change was discussed in a public forum, but at a high level and without consideration of potential abuse: https://adblockplus.org/forum/viewtopic.php?p=179030.

Domain	AdGuard Reduction	ABP Reduction
doubleclick.net	28%	41%
googlesyndication.com	3%	37%
pubmatic.com	0%	52%
adnxs.com	0%	0%
openx.net	0%	46%
rubiconproject.com	0%	46%
2mdn.net	1%	0%
yandex.ru	100%	100%
amazon-adsystem.com	0%	21%
bidswitch.net	100%	100%

Table 4.4: **Exchange Comparison on Easylist Variants**— Here we show the reduction of blocking the top ad exchanges from Easylist and Easyprivacy for the Ad Blockers. Both ad blockers show significantly reduced blocking of most ad exchanges, even though they are based upon Easylist. Neither ad blocker incorporates Easyprivacy. Percentages indicate the percent of requests blocked by EasyList and EasyPrivacy that were not blocked by the tested tool (0% indicating no change and 100% indicating no blocking).

are to allow cookie matching behavior by loading empty content or tracking pixels. AdBlock Plus allows more of these tracking pixels to proceed from a more diverse set of sources.

4.5 Blocking Completeness

Most current blocking extensions build their rule sets of what to block through manual effort. In this section we quantify the completeness of these efforts for different blocking extensions. Naively, we could look at all known advertisers and treat this as each extensions' ideal, however this fails to account for the different subsets of the A&A ecosystem each extension wishes to block. Instead, we identify ad exchanges blocked by each extension and label the eFLDs that receive at least 10% of their connections from these extensions as advertisers for that extension.

Prior work using inclusion graphs has identified ad exchanges using heuristics on the ratio of indegree and outdegree and outdegree alone of nodes in the A&A inclusion graph. Ad exchanges must have large outdegree to other advertisers to be effective. They also have approximately equal indegree and outdegree. Too large an indegree indicates a tracker that only ingests information without selling impressions, while too small an indegree indicates an SSP that is not purchasing

Domain	Out-degree	In/Out-degree Ratio
doubleclick.net	585	1.22
adnxs.com	449	0.66
googlesyndication.com	274	1.48
rubiconproject.com	254	0.57
openx.net	200	0.55
demdex.net	193	0.70
bidswitch.net	191	0.61
pubmatic.com	191	0.75
mathtag.com	189	0.58
adform.net	170	1.01
amazon-adsystem.com	163	0.67
criteo.com	153	0.64
turn.com	142	0.52
yahoo.com	140	0.78
taboola.com	137	0.75
crwdcntrl.net	133	0.70
bluekai.com	129	0.76
lijit.com	129	0.93
krxd.net	127	0.76
contextweb.com	127	0.68
smartadserver.com	122	0.65
w55c.net	105	0.52
everesttech.net	103	0.64
1rx.io	100	0.66
tapad.com	94	0.74
3lift.com	86	0.74
gumgum.com	86	0.78
spotxchange.com	80	0.51
serving-sys.com	79	1.95
wp.com	75	1.52
2mdn.net	73	1.51
teads.tv	71	0.86
imasdk.googleapis.com	68	1.40
outbrain.com	68	0.56
sharethrough.com	66	0.68
yandex.ru	66	0.76
adition.com	62	1.05
sonobi.com	62	0.77
addthis.com	62	1.15
rfihub.com	59	0.69
sharethis.com	59	0.71
eyeota.net	58	0.55
stickyadstv.com	58	0.66
yimg.com	57	0.96
33across.com	57	1.19
media.net	56	0.89
bttrack.com	51	0.65

Table 4.5: **Ad Exchanges**— Ad exchanges, as determined by Section 4.5 are shown here, with the parameters used to classify them. These parameters are the out degree and ratio of in degree to out degree in the Inclusion graph including only nodes that are advertising and analytics domains.

ads from a large number of platforms. We look at the values of outdegree and degree ratio for known ad exchanges and determine that an outdegree ≥ 50 and in/out degree ratio $\in [.5, 2.0]$ was

representative. For EasyList and EasyPrivacy, this labeled 47 ad exchanges, enumerated in Table 4.5. This is performed over the subgraph of our inclusion graph that only includes A&A eFLDs.

With ad exchanges labeled for each extension, we then identify the eFLDs that are advertisers. We note that we labeled eFLDs as advertising or analytic if they were blocked more that 10% of the time. Since connections to ad exchanges would be blocked, preventing them from including any further domains, we look at the percent of connections each eFLD receives from ad exchanges. Therefore, we label all eFLDs with at least 10% of their connections coming from ad exchanges as advertisers. The results of this process are show in Table 4.6.

Note that while this quantifies coverage of the block lists, a lack of coverage does not inherently mean that privacy is adversely affected. We found these advertisers by identifying domains that *would have been blocked* by the tool in some cases by virtue of blocking ad exchanges. However, blocking the advertisers themselves can be useful to block their direct inclusion from publishers or otherwise unblocked members of the A&A ecosystem.

In Table 4.6 we see that the lowest coverage is observed by the smallest lists. These lists, on manual inspection, seem to focus on the larger players in the analytics ecosystem and are focused in their mission to provide privacy. A focused effort on the actors that share information the most seems to be an efficient use of manual effort; in Section 4.6 we will quantify the efficacy of this effort on privacy.

The best coverage of domains is provided by uBlock Origin. uBlock Origin curates its block list

List	Known A&A eFLDs	Discovered A&A eFLDs
AdBlock Plus	2,545	3,853 (151%)
AdGuard	2,981	3,893 (131%)
Disconnect.me	1,340	4,397 (328%)
DuckDuckGo	943	3,277 (347%)
Ghostery	2,190	5,397 (246%)
uBlock Origin	6,115	4,289 (70%)

Table 4.6: **List Coverage Quantification**— We show the results of quantifying missed A&A subdomains over the tested extensions using our exchange in-degree metric from Section 4.5. Note that this does not quantify privacy loss; this does not account for the blocking that prevents further requests from being performed.

through a large community effort. This effort shows in the size and diversity of eFLDs blocked by its list as shown in Table 4.2.

4.6 Privacy Impacts

One aspect of the comparison of different blocking extensions in Section 4.4 is evaluating the number of blocked requests in our web crawl. While this does provide a rough proxy for privacy offered by the extension over the crawl, it is inaccurate for two main reasons. First, it does not evaluate how much of the tracking information is shared with each individual domain or out-of-band information sharing, such as that performed by ad exchanges. Second, it does not account for the impact of blocking one request that would otherwise lead to many other requests. We use an approximation of what impressions are learned by each advertiser to take these factors into consideration and more accurately reflect privacy violations.

4.6.1 Exchange Sensitivity

While we believe we understand the A&A ecosystem well enough to identify the ad exchanges, we wish to understand the impact of this classification on our privacy metrics. Our classification depends upon the ad hoc selection of thresholds of outdegree and in/outdegree ratio, so we desire more context of the impact this makes on privacy.

To show the effect this selection has on privacy, we sweep the number of ad exchanges we include from 0 up to the 47 selected in Section 4.5, always adding the origin that connects to the most distinct A&A domains. We graph the Nth percentile ad exchange's visibility into site impressions as we vary the number of exchanges. The results of this are shown in Figure 4.3.

The most impactful ad exchanges on the privacy of end-users are generally the largest, which we add first in our sweep. We observe this by the majority of growth in each percentile we show occurring over the first few exchanges we add. Beyond this point, each exchange has very little impact on the number of impressions observed by each A&A domain. This is consistent with prior work and intuition: those agents that share information most broadly are those most impactful to privacy.

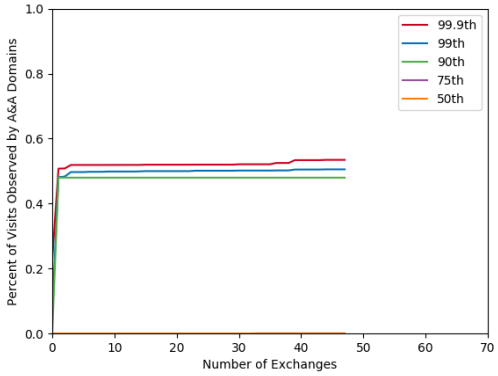


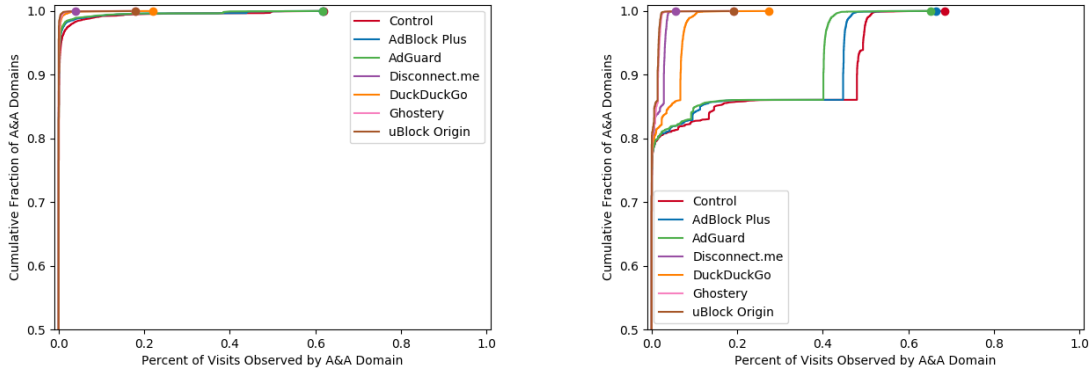
Figure 4.3: **Ad Exchange Count Sweep**— We show the degree of sensitivity to the number of ad exchanges on various percentiles of vist leakage. We add brokers in the order of the number of advertisers the initate connections to in our web crawl. Each line represents some Nth percentile of success in observing as many page visits as possible over our web crawl.

4.6.2 Tool Evaluation

With inclusion graphs for each site impression and the blocking behavior of six different tools captured, we can compare the privacy granted through the use of each tool. We add a seventh measurement to our six tools we analyze in previous sections: the control, in which we perform no blocking. Further details of how we perform this analysis and both models of information sharing are described in Section 4.3.5. The distributions of information sharing under these models are shown in Figure 4.4.

In the Cookie Matching model there is much less impression disclosure. Any blocking performs better than nothing, however the advertising focused services (AdGuard and AdBlock Plus) provided very little benefit in blocking the worst case A&A domains. The remaining blocking tools performed similarly better with Disconnect.me having the clearly best worst-case performance. In the Real-Time Bidding model we see similar trends, but with overall more disclosure as exchanges provide more sharing.

In the Real-Time Bidding model, we observe substantial improvements in user privacy by some extensions, and similarly we see the least improvement in tools focused on advertisement blocking. However, there is no uniformly best extension for user privacy: Disconnect.me again has the best



(a) Cookie Matching

(b) Real-Time Bidding

Figure 4.4: **Privacy Evaluation of Blocking Extensions**— We show the rate of visit disclosure over our known advertisers using two different disclosure models. At the left we see the Cookie Matching model and at right we see the Real-Time Bidding model.

worst case performance, but lags Ghostery and uBlock Origin in the 85th to 99th percentile of A&A domains. Depending on threat model, concern with the most privileged A&A domain, specific A&A domains, or average performance may be used to select the defense among extensions focused on privacy. This figure the used to generate it are needed to answer those questions quantitatively.

Tools have similar classes of performance grouped based upon their mission and similar order of worst case performance in both cases. It is unclear whether there is a universal best strategy independent of threat model and user behavior. However, tools with an emphasis on privacy show large improvements to user privacy.

Overall, several aspects of these distributions are interesting in comparison to prior work. The rates of observation by publishers are lower, providing a more optimistic view of third-party tracking than other studies. There are several possible explanations for this: differences in how these rates of observation are computed, differences in the population of sites being measured, and differences in how sites are sampled from this population. For example, we do not focus our study on sites that have a higher incidence of tracking techniques, e.g. shopping sites. Also, as shown in Figure 4.5, ad exchange prevalence is reduced in sites beyond the top thousand, where prior work has focused.

However, we see similar qualitative differences between the Cookie Matching model and Real-

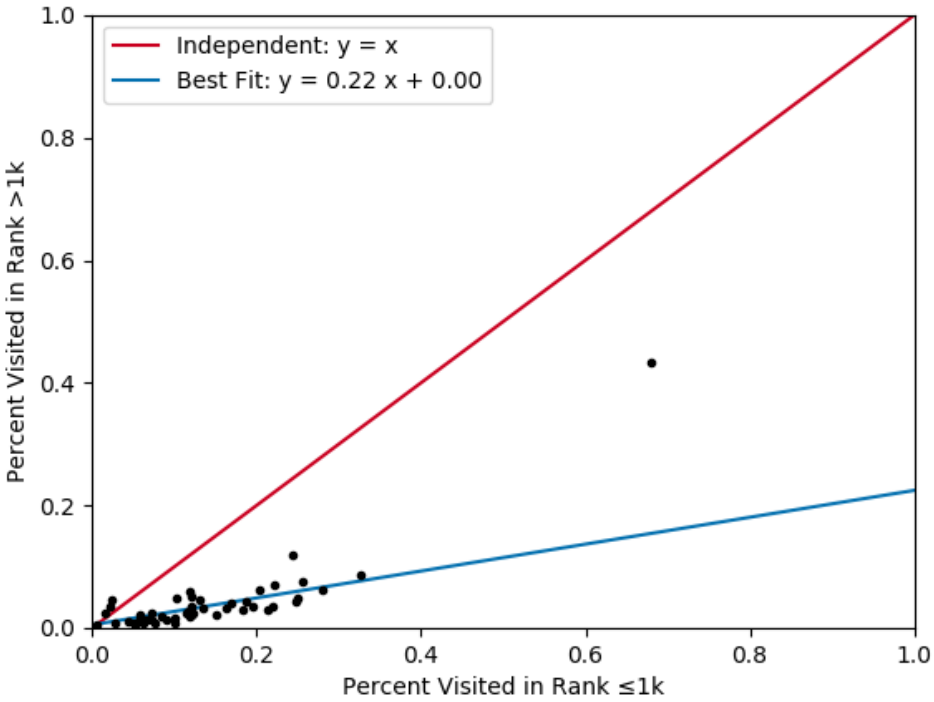


Figure 4.5: **Exchange Connection Frequency by Rank**— We demonstrate on a scatter plot with best fit line that ad exchanges generally are five times as likely to be on a website in the top thousand than in the rest of the top million domains. This is with the removal of the outlier `doubleclick.net`, at the upper right of the plot.

Time Bidding model, described in Section 4.3.5, as the prior work that introduced them. Under Cookie Matching, very few A&A parties have significant access to the users' browsing patterns. Under Real-Time Bidding, there are some number of A&A domains that observe almost no traffic, and a wide variety of scopes of observation for the most well positioned A&A domains.

4.6.3 Acceptable Advertisement

In Section 4.4.2 we observed that only AdBlock Plus has any change in blocking behavior when the rules explicitly allowing domains are removed. It is also the worst performing blocking extension. We know that some privacy is lost due to the Acceptable Advertising list, and prior work has identified it as a major source of reduced efficacy. However, we find this not to be the case under our more accurate representation of the filter lists. Figure 4.6 shows the effect of removing the Acceptable Advertising list from AdBlock Plus.

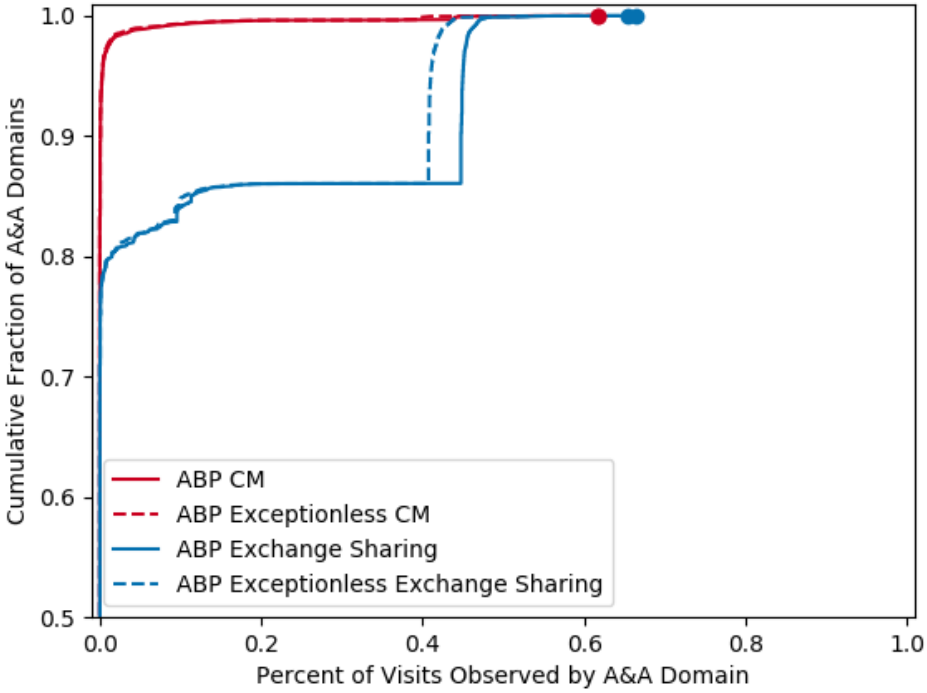


Figure 4.6: **Evaluating Acceptable Advertisements**— We show the impact of the Acceptable Advertising list on the privacy of AdBlock Plus in our crawl over both models of information sharing.

	Cookie Matching		**Real-Time Bidding**	
	Average	Worst	Average	Worst
Control	0.238%	36.90%	4.475%	40.88%
AdBlock Plus	0.183%	36.78%	4.063%	39.60%
AdGuard	0.162%	36.75%	3.667%	38.88%
Disconnect.me	0.015%	2.30%	0.309%	3.38%
DuckDuckGo	0.028%	13.10%	0.718%	16.38%
Ghostery	0.016%	10.70%	0.181%	11.43%
uBlock Origin	0.011%	10.65%	0.158%	11.45%
AdBlock Plus (Exceptionless)	0.166%	36.78%	3.720%	38.98%

Table 4.7: **Quantified Privacy Impact**— We provide here the numerical values achieved for our privacy evaluation experiments in Section 4.6. The percent of visits disclosed averaged over all know A&A domains and to the most priveleged A&A domain are provided for both information sharing models we considered. This is a tool to clarify details that may be obscured by overlapping lines or hard to visually compare, such as area under the curve. We note that overlapping lines indicate that we did not observe a significant difference.

We provide a quantified comparison of all analyses in this Section in Table 4.7 to assist in reading features of line charts that may be challenging to read, such as area under the curve.

4.6.4 Improved Model

In Section 4.3.2, we noted that our model of blocking rules is more detailed than that of prior work, and that allows us to better understand the role of acceptable advertising programs. To justify this, we repeat our experiments from Section 4.6.3 without consideration for rule flags, to show how results change under a simplified model like that of prior work. This is shown in Figure 4.7.

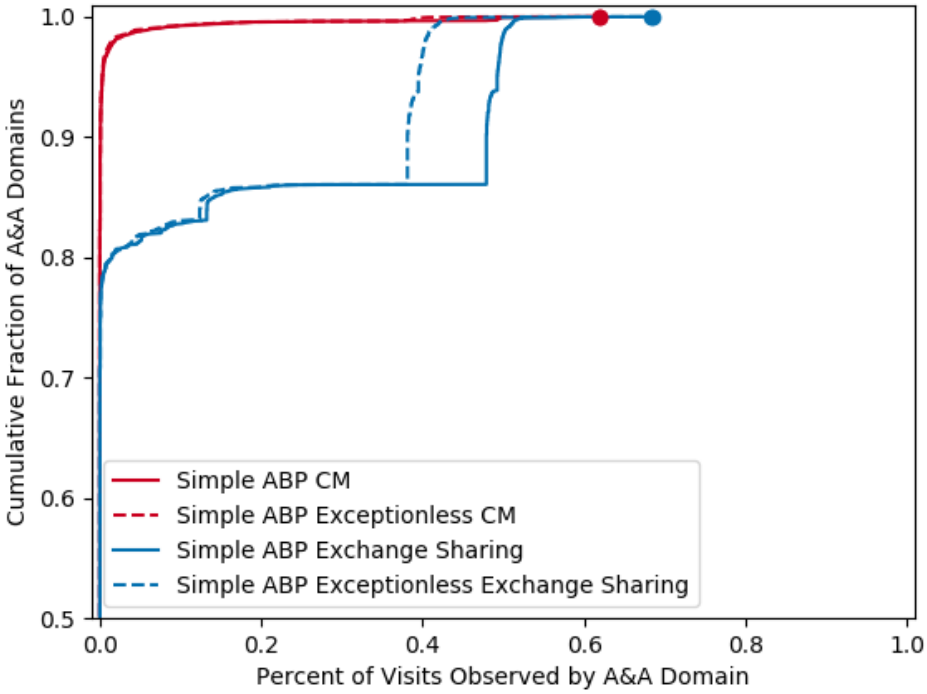


Figure 4.7: **Evaluating Acceptable Advertisements Under Simplified Model**— We show the impact of the Acceptable Advertising list on the privacy of AdBlock Plus in our crawl over both models of information sharing when the rule flags are not considered, as in prior work. The impact of the Acceptable Ads program is exaggerated.

The effects of the acceptable advertising program is exaggerated without the rule flags. With flags removed, the tool performs even better without exceptions and even worse with exceptions. Particularly notable is that when we do not consider rule flags, we reproduce, from Bashir et al., that AdBlock Plus performs very similarly to the control. This is due to the inclusion of many

ad exchanges in the exception list that have ignored domain constraints. We note that in our experiments, the difference between AdBlock Plus without rule flags and no blocking exists, but is within the confidence interval our sample size allows.

4.7 Discussion

In this chapter, we have quantified several aspects of popular blocking extensions. We found considerable differences between the extensions and a few surprising similarities. Our results lead us to a few key lessons, caveated by the limitations of this work.

4.7.1 Findings and Lessons

Several key lessons can be drawn from our results. First, we found that extensions focused on ad blocking (AdGuard and AdBlock Plus) provide the weakest practical protection for users. This is despite using privacy as a marketing tactic on their websites. Second, we found that despite building off of EasyList and EasyPrivacy to build their block lists, uBlock Origin, AdGuard, and AdBlock Plus all behave very differently in what they block and how much protection they provide the end user. Third, we found evidence that while AdBlock Plus has the only significant Acceptable Advertisement program, it only accounts for a 9% increase in total impression disclosures in the Real-Time Bidding model. Finally, we find that completeness of manual effort to enumerate advertisers to block does not correspond to better user privacy; Disconnect.me has the worst measured completeness and has strong privacy properties. From these findings, we can draw the following lessons.

If users seek a blocking extension that benefits their privacy, they should focus on those whose primary emphasis is privacy or user agency. In particular, Disconnect.me provides the best protection against the very strongest web trackers and uBlock Origin and Ghostery provide slightly better protection against the next thousand best web trackers. If users are comfortable configuring their extension with custom block lists, they may not even have to make this tradeoff.

We also learn that distributed manual effort is not needed to curate an effective block list. The roughly 500 domains identified and blocked by Disconnect.me provided some of the best coverage

we saw. A focus on continued identification and third-party blocking of major players in the advertising ecosystem is likely to provide the best trade-off of effort for privacy. Additionally, this may help future performance optimization of tracker blocking tools. This is likely because of the significant presence and interconnectedness a few players have in this ecosystem.

Finally, we learn that despite their marketing, and even with Acceptable Ads programs disabled, ad blockers are not nearly as effective privacy enhancing technologies as their peers that focus on user privacy and agency.

4.7.2 Limitations

Our work presents an improved understanding of the Advertising and Analytics ecosystems over state of the art. However it is important to understand where the potential shortcomings of our model fall so that we can both understand our results in the appropriate context and understand avenues of future work.

For example, our implementation of blocking rule application implements more features than other work that were critical to understanding the role of Acceptable Advertising programs [23]. However, we do not implement all documented features, let alone capture any possible undocumented features of each tool. While this may reduce our accuracy, we believe our improvement and the insight it provides is significant.

Additionally, our results driven by our crawl data are restricted to the sites we evaluate: home pages of the Tranco top million sites. However, we add to the literature an updated million site measurement measurement not drawn from proprietary data sources.

We do not evaluate the quality of the candidate advertisers we discover through our heuristics. It is an open problem to automatically differentiate between "breaking" a site such that it is no longer functional and effectively blocking A&A techniques. Current promising approaches are browser instrumentation of what sites cause users to disable their blocking extension. We consider this evaluation out of scope, and believe that our small candidate lists are currently useful for several purposes: manual evaluation for current browser extensions and a test-list with low false positive rate to reduce base-rate concerns when automatically adding them to browsers to be evaluated.

Finally, we acknowledge that our results on user privacy are based on a model of information sharing that is an approximation of one aspect of the A&A ecosystem. The model does not capture all active members of the ecosystem or information channels that do not involve interaction with the end-user machine. For example, first-parties collecting information for sale is missed in our models.

4.8 Conclusion

The insight we gain into blocking extensions from this work will help further improvement of these blocking extensions and privacy advice surrounding them. We are able to identify similarities between tool behavior that indicate list sharing. We use these lists to identify ad exchanges and candidate advertisers using simple heuristics of the inclusion graph. We discover that only AdBlock Plus has a significant explicit exception program. We find that these features are not just true of the rules exercised by our crawl, but also of the rules defined overall.

We identify privacy impacts of browser extensions by analyzing how each extension affects which parties receive requests from page visits in our crawl. This allows evidence driven recommendations to use Disconnect.me, Ghostery, or uBlock Origin to users and support for Firefox's selection of the Disconnect.me list for its content blocking deployment. The differences we observe also allow inference that organizational mission plays the largest part in determining an extension's efficacy against third party tracking under default configurations. Also, we confirm that list size is not a meaningful proxy for provided privacy.

These results comprise a significant improvement in our understanding of blocking extensions and our ability to aid in the defense of user privacy from third-party web tracking.

CHAPTER V

Measurement of HTTPS Certificates

At the root of all secure systems is a trusted element. For the web, the trusted element that allows security properties to be built upon is the certificate ecosystem. Certificates link domain names to cryptographic keys, allowing a user-agent to reason about the authenticity and confidentiality of the information being exchanged. However, certificates have been mis-issued in the past: an attacker obtains a certificate linking a key they own with a domain that they do not. This is a problem for everyone relying upon web security. Users may be uncertian if the certificate they receive has been misissued, web admins do not know if the only certificates for their site are correctly issued, and certificate authorities can not know if they were tricked into misissuing a certificate. Transparency into the certificate ecosystem can improve the trust we place in certificates.

Our work is the first to comparatively analyze different views of the certificate ecosystem to understand the extent of their views, and therefore utility in extending trust. We find that the certificate transparency project from Google had made bounds forward in improving transparency, but failed to capture significant portions of non-web TLS servers. Moreover, we find a significant shortcoming in IPv4 scanning in assessing the diversity of web servers. We merge together complementary data sets of certificate transparency and Censys to make it easier for web admins to check for mis-issued cetificates. These combined data sets remain searchable for free, and are still continuously synchronized four years after the completion of the initial project.

This chapter is adapted from a joint publication that first appeared in the Proceedings of the

2016 Internet Measurement Conference (IMC'16) [198].

5.1 Introduction

Nearly all secure web communication takes place over HTTPS. Both the underlying TLS protocol and the supporting certificate public key infrastructure (PKI) have been studied extensively over the past five years, with questions ranging from understanding the behavior of certificate authorities [58, 65] to detecting server-side vulnerabilities and tracking how quickly they are mitigated [53, 57, 191].

Such measurements are difficult to conduct well, since there is no comprehensive set of trusted certificates or of HTTPS websites—no ground truth for studying this ecosystem. Instead, researchers have attempted to gain visibility into it using various fragmentary perspectives—such as scanning the IPv4 address space [59], querying popular Alexa domains, passively monitoring network traffic [13], and querying Certificate Transparency (CT) logs [114, 124]. Each methodology provides an imperfect view of the world, yet there has been little work to analyze how they differ or how they might be combined to piece together a more comprehensive picture.

Consider, for example, the different perspectives provided by CT logs and the Censys search engine [55], two widely used sources of certificate data. CT is designed to enable auditing of trusted certificates by recording them in publicly verifiable logs. While this may someday provide a complete view of the certificate ecosystem, at present, publishing certificates to CT logs is voluntary in most cases. In contrast, Censys provides a public database of certificates collected by actively scanning the IPv4 address space and Alexa Top Million domains. Although IPv4 scanning might seem to promise an exhaustive view of certificates in use on the public Internet, it misses several important cases, including those served exclusively over IPv6. IP-based scanning also cannot provide the TLS Server Name Indication (SNI) header [61], which specifies the requested domain name and is necessary when a server hosts multiple sites from a single IP address.

In this work, we comparatively analyze the certificates seen by eight measurement perspectives: (1) a Censys certificate snapshot, (2) an exhaustive IPv4 scan on TCP/443, (3) a scan of Alexa Top

Perspective	Valid Certificates	Exclusive Certificates	FQDNs	Sites	Description
CT Logs	15,374,936	6,830,849	29,967,065	12,202,712	Certificates in the well known CT logs
Censys Snapshot	6,448,588	609,773	14,495,436	4,817,530	Database of IPv4 and Alexa Top 1M scan results
CT Scan	6,419,584	54,225	20,967,351	9,153,162	Scan of all FQDNs in CT logs
IPv4 HTTPS Scan	3,760,360	637	8,209,465	2,926,766	HTTPS scan of all IPv4 addresses
`.com`, `.net`, `.org` zones	2,436,425	31,195	11,892,649	5,669,024	Root and `www.` names in `.com`, `.net`, and `.org`
Common Crawl	1,258,886	1,154	6,177,985	2,885,466	Scan of domains crawled by Common Crawl
Alexa Top 1M	288,670	0	1,907,256	917,259	HTTPS scan of Alexa Top Million domains
ICSI Notary	256,869	3,805	2,040,138	916,612	Observed in passive network traffic inspection
"Universe"	16,989,236	—	32,454,062	12,673,515	Certificates in any of the perspectives above

Table 5.1: **Certificate Perspectives**—We compare eight distinct perspectives on the universe of valid certificates, spanning six measurement techniques. Certificate Transparency (CT) is the largest perspective, including many certificates only seen in CT.

Million domains, (4) a snapshot of public CT Logs, (5) a scan of domains contained in these CT logs, (6) a scan of domains contained in the `.com`, `.net`, and `.org` zone files [205], (7) a scan of domains from the Common Crawl dataset [43], and (8) certificates passively observed by the ICSI SSL Notary using passive network monitoring [13].

Combining these datasets, we observe nearly 17 million unique browser-trusted certificates that were valid during our measurement interval, August 29 to September 8, 2016. Of these, 90.5% appeared in public CT logs and 38.0% were seen by Censys. To understand this difference, we investigate the impact of SNI by attempting connections to 30 million domains extracted from certificates in CT logs. Only 35% of domains that accepted a connection with SNI offered the same certificate when SNI was not used. This places an upper bound on the certificates observable by IP-based scanning. Combining data from Censys and CT covers 99.4% of all trusted certificates seen by any perspective we studied, and may closely approximate the public HTTPS ecosystem. However, as the vast majority of the certificates in our data originate from these two sources, this number is suspect. To better validate the fraction of certificates visible with these perspectives, we consider certificates seen by scanning domains from the `.com`, `.net`, and `.org` zone files, and find that the union of CT logs and Censys contains 98.5% of them.

Based on these results, we recommend that researchers performing future HTTPS measurements use a combination of data published in CT logs and Censys-style IPv4 scanning. To facilitate this, we are working with the operators of Censys to implement synchronization between Censys and CT

logs. Going forward, Censys will continuously incorporate certificate data from public CT logs in its results and publish newly discovered certificates back to Google CT logs, making either data source a strong foundation for studying the certificate ecosystem.

5.2 Certificate Perspectives

In order to compare techniques for measuring certificates, we conducted six kinds of scans and analyzed two existing datasets. Table 5.1 summarizes these perspectives, which we describe in detail below.

This study is the first to comparatively analyze multiple passive and active datasets related to TLS. With the current use of Internet Measurement to understand the scope and importance of vulnerabilities in TLS, it is vital to understand the biases and limitations of these datasets. As such, we use techniques common to TLS measurement generally. We additionally pursue a few techniques that are, at time of original publication, unused in Internet Measurement of TLS.

5.2.1 Perspectives Enumerated

Certificate Transparency Logs Certificate Transparency (CT) aims to allow public auditing of trusted certificates [124]. Anyone can submit valid certificates to CT log servers, which record them in cryptographically verifiable public ledgers. Although there is no universal requirement for submission, Google records all certificates seen in its web crawls to CT logs. Chrome requires all issuers submit extended validation (EV) certificates to at least two logs [36]. Chrome recently mandated that Symantec certificates signed after June 1, 2016, be submitted to be trusted as well [182]. Several CAs voluntarily log all certificates they issue, notably Let's Encrypt [130] and StartCom [184].

We retrieved the certificates stored in twelve well-known CT logs on September 8, 2016. These logs are operated by Google ("Pilot", "Aviator", "Rocketeer", and "Submariner"), Digicert, StartCom, Izenpe, Symantec, Venafi, WoSign, CNNIC, and Shengnan GDCA.

Censys Certificate Snapshot The Censys search engine [55] publishes daily snapshots of all the certificates it indexes. Censys collects certificates by exhaustively scanning the IPv4 address space

without SNI, and by connecting to all Alexa Top Million domains with SNI. Our perspective is based on the September 8, 2016 snapshot.

Scan of FQDNs from CT We extracted the fully qualified domain names (FQDNs) from all certificates in our CT log snapshots, covering the common name (CN) and subject alternative name (SAN) fields. We then used ZGrab [55] to attempt an HTTPS connection to each domain, with SNI enabled. The scan ran from the University of Michigan on August 29 and September 6 and 8, 2016.

IPv4 HTTPS Scan We used the ZMap suite [59] to scan the IPv4 address space for HTTPS servers listening on TCP/443. The scan took place on August 29, 2016, from the University of Michigan. For each listening host, we attempted a TLS handshake and recorded the presented certificate chain. Since these connections were based on IP addresses rather than domain names, they did not include the SNI header.

Authoritative Zone Files We attempted HTTPS handshakes with all domains in the authoritative zone file [205] for `.com`, `.net`, and `.org` domains, for both the base domain and the `www` subdomain. (Since the TLD zone files contain only the name server entries for each domain, we learn only the base domain name.) We ran these scans using ZGrab on August 29 and September 2, 2016, from the University of Michigan. There were 153 million unique domains in these zone files, and we completed 42 million successful HTTPS handshakes to the base domains, and 40 million successful HTTPS handshakes to the `www` subdomains. While we connected to many domains, the certificates served were often only valid for the hosting provider's domain name and not the scanned domain.

Common Crawl The Common Crawl project [43] aims to perform a regular, complete crawl of public websites. We processed the January 2016 crawl and extracted 28.9 million unique domains. We used ZGrab to attempt an HTTPS connection to every domain on September 3, 2016, from the University of Michigan.

Alexa Top Million HTTPS Scan We used the ZMap suite to attempt connections to the Alexa Top Million domains. The scan took place on September 3, 2016, from the University of Michigan. For each listening host, we attempted a TLS handshake with SNI enabled and recorded the presented

certificate chain.

ICSI SSL Notary The SSL Notary dataset consists of daily Internet traffic from approximately 180,000 users at five North American academic or research institutions [13]. We analyzed 2.2 billion TLS connections on TCP/443 from July 29 to August 29, 2016, extracting a total of 635,314 certificates. We excluded incomplete connections as well as HPC, Grid, and Tor certificates, resulting in 386,051 certificates, of which 256,869 were trusted by the Mozilla NSS root store.

Due to a nondisclosure agreement that limited our internal data sharing, Notary certificates are not included in cases where we consider the union of all perspectives. This reduces the size of the union by 0.02%.

In total across all these perspectives, we discovered 17 million unique certificates that were valid and trusted by the Mozilla NSS root store. Since the different datasets contain somewhat different temporal perspectives, we consider certificates to be valid only if their date ranges cover our entire collection period, August 29 to September 8. By constraining our data in this way, we ensure that no data source contains certificates that would be invalid in another data source due to the time when the certificates were validated.

5.2.2 Ethics of Measurement

For our active scanning, we honored the University of Michigan's institutional blacklist to exclude endpoints that previously requested not to be scanned. We also followed the best practices defined by Durumeric et al. [59]; we refer to that work for more discussion of the ethics of active scanning. Passive data collection was cleared by the responsible parties at each contributing institution. The ICSI SSL Notary stores connection metadata (e.g., certificate and cipher information) without collecting any connection payload.

5.3 Impact of Perspective on HTTPS Research

A number of recent studies have used IPv4 and Alexa Top Million scanning to measure how HTTPS is deployed in the wild [25, 53, 55, 57, 58, 65, 127, 191, 206, 218]. Our finding that IPv4 scans miss nearly two-thirds of certificates suggests that, if these studies were performed today, they

might not accurately reflect the state of the Internet.

To provide a concrete example of the differences caused by different perspectives, we present a survey of sites vulnerable to the FREAK attack [191] in Table 5.2. The data comes from scanning each of: the IPv4 address space, CT log FQDNs, Alexa Top Million domains, our zone files, and domains extracted from Common Crawl. For each set, we measure how many responsive hosts are vulnerable.

The number of vulnerable hosts changes with each perspective we measure. The vulnerability rates range from 1.88% of IPs vulnerable when measured by IPv4 scanning, to 4.54% of IPs vulnerable when measured by our Common Crawl scan. This variation demonstrates the necessity of considering the perspective used when performing measurements.

We also observe differences when comparing the most common certificate issuers seen in each perspective, as shown in Table 5.3. We measure this by grouping certificates by their issuer organization field and manually deduplicating similar issuer names. Different perspectives display differing views on which CA is most popular. For example, Let's Encrypt is the most popular CA in CT, Common Crawl, and zone scans, but it is ranked fifth in IPv4 scans.

5.4 Two Complementary Perspectives Emerge

Our certificate "universe" consists of 16,989,236 unique, valid certificates from our eight perspectives. These certificates contain 32,454,062 FQDNs from 12,673,515 sites (as defined by the public suffix list [146]). The CT logs and Censys snapshot are the two largest datasets. The CT logs contain 15,374,936 certificates (90.5% of certificates observed in this study). Censys sees 6,448,588 certificates—38% coverage of the certificates observed in this study. When combined, these two perspectives provide 99.4% coverage of all certificates we observe and 99.7% coverage of sites.

5.4.1 Limitations of Censys

The Censys snapshot only covers 38% of our certificate universe. Two sources are responsible for approximately 64% of the missing certificates. As shown in Table 5.5, 90.8% of Let's Encrypt

	IPv4 Addresses	FQDNs	Sites
IPv4 Scan	1.88%	4.14%	5.18%
Common Crawl	4.54%	1.42%	1.51%
Zones	3.91%	0.90%	0.96%
Alexa	2.71%	0.90%	0.96%
CT Scan	3.73%	1.12%	1.18%

Table 5.2: **Rate of Vulnerability to FREAK**—Vulnerability rates measured by each methodology vary significantly.

Alexa Top 1M		Common Crawl		CT Scan		IPv4 Scan		Zone File	
Comodo	30%	Let's Encrypt	24%	Let's Encrypt	41%	Comodo	17%	Let's Encrypt	28%
GeoTrust	17%	Comodo	21%	Comodo	15%	GoDaddy	16%	Comodo	24%
GoDaddy	10%	GeoTrust	15%	GeoTrust	9%	GeoTrust	15%	GeoTrust	12%
Let's Encrypt	8%	GoDaddy	9%	GoDaddy	7%	GlobalSign	7%	GoDaddy	12%
GlobalSign	7%	cPanel	7%	GlobalSign	5%	Let's Encrypt	7%	cPanel	7%
Other (351)	29%	Other (475)	25%	Other (560)	23%	Other (567)	62%	Other (377)	18%

Table 5.3: **Top Certificate Issuers**—The most common issuers differ depending on the perspective studied. Let's Encrypt has its highest popularity in CT Logs, cPanel only appears in the top five for zone file scanning, and IPv4 scanning yields a longer tail.

certificates are absent from Censys, accounting for 42% of the certificates missed by Censys and 23.5% of the certificate universe. Let's Encrypt submits all issued certificates to CT, but it appears that many of these certificates are inaccessible without SNI or are not served on public sites.

CloudFlare accounts for 17% of all certificates in this study, but Censys misses 81% of these, resulting in an exclusion of 13.9% of the certificate universe. We manually confirmed that the vast majority of CloudFlare certificates are only accessible through SNI. This intuitively makes sense because most Censys certificates are found through IPv4 scans that do not include SNI information.

5.4.1.1 Server Name Indication

In order to directly measure the impact of SNI, we performed two scans over all FQDNs contained in valid certificates in the CT logs. We scanned 30 million domain names, and were able to complete successful HTTPS handshakes with 68% using SNI. As shown in Table 5.6, only 77% of domains that accepted a connection made with SNI accepted connections without it, and only 35% returned the same certificate as when SNI was used. This further shows that scanning without SNI misses a substantial fraction of websites.

	Certificates missing from CT	Fraction missing	Fraction of universe
All	1,614,300 (100.0%)	9.5%	9.5%
GoDaddy	314,966 (19.5%)	29.6%	2.0%
cPanel	120,907 (7.5%)	28.4%	0.7%
Thawte	56,078 (3.5%)	18.0%	0.3%
Starfield	38,220 (2.4%)	34.5%	0.2%
Other	1,084,129 (67.2%)	7.2%	6.4%
`mail`	188,109 (11.6%)	20.8%	0.7%
`*`	142,303 (8.8%)	16.0%	0.8%
`vpn`	32,377 (2.0%)	50.8%	0.2%
`www`	147,588 (9.1%)	4.3%	0.9%
Other	1,131,862 (70.1%)	9.3%	6.7%

Table 5.4: **Certificates Missing from CT**—Some issuers, such as GoDaddy, have their certificates appear in CT at a lower rate than the general population, and the same is true of `mail.`, `*.`, and `vpn.` subdomain certificates.

	Certificates missing from Censys	Fraction missing	Fraction of universe
All	10,540,648 (100.0%)	62.0%	62.0%
Let's Encrypt	4,401,674 (41.8%)	90.8%	23.5%
CloudFlare	2,381,940 (22.6%)	81.3%	13.9%
Other	3,757,050 (35.6%)	40.7%	22.1%

Table 5.5: **Certificates Missing from Censys**—Let's Encrypt reports all certificates to CT, even those never served. Cloudflare certificates are only served with SNI.

		FQDNs from CT
With SNI	Accepted connection	20,305,155 (100%)
Without SNI	Accepted connection	15,598,532 (77%)
	Same certificate as SNI	7,021,206 (35%)
	Different certificate	8,577,326 (42%)

Table 5.6: **SNI Behavior**—77% of active domains extracted from CT logs accepted connections without SNI, but only 35% served the same certificate as when contacted with SNI.

# seen in Alexa Top 1M		... in both CT and IPv4 scans	... in CT scan only	... in IPv4 scan only	... in neither
Certificates	288,220	203,842 (70.7%)	76,085 (26.4%)	7,236 (2.51%)	1,057 (0.37%)
FQDNs	1,906,302	663,129 (34.8%)	1,229,484 (64.5%)	11,769 (0.62%)	1,920 (0.10%)
Sites	916,789	327,894 (35.8%)	583,474 (63.6%)	4,663 (0.51%)	758 (0.08%)

Table 5.7: **Coverage of Alexa Results from CT Scans and IPv4 Scans**—IPv4 scanning misses 26% of certificates, 65% of FQDNs, and 64% of sites found in our Alexa scans, but combining IPv4 and CT scans yields >99% coverage for each category.

# seen in any perspective		... in both CT logs and Censys	... in CT logs only	... in Censys only	... in neither
Certificates	16,989,236	4,927,174 (29.0%)	10,447,762 (61.5%)	1,521,414 (8.96%)	92,886 (0.55%)
FQDNs	32,454,061	12,156,237 (37.5%)	17,817,430 (54.9%)	2,379,463 (7.33%)	100,931 (0.31%)
Sites	12,673,514	4,412,464 (34.8%)	7,794,002 (61.5%)	426,168 (3.36%)	40,880 (0.32%)

Table 5.8: **Coverage of Universe in CT Logs and Censys**—Combining the results from CT logs and Censys covers more than 99% of the certificates, FQDNs, and sites that can be found using any of our perspectives.

# seen in our zone scan		... in both CT logs and Censys	... in CT logs only	... in Censys only	... in neither
Certificates	2,431,246	1,283,379 (52.8%)	1,073,965 (44.2%)	37,825 (1.56%)	36,077 (1.48%)
FQDNs	11,881,085	5,231,678 (44.0%)	6,495,535 (54.7%)	65,664 (0.55%)	88,208 (0.74%)
Sites	5,663,431	2,544,950 (44.9%)	3,054,848 (53.9%)	26,409 (0.47%)	37,224 (0.66%)

Table 5.9: **Coverage of Zone Results from CT and Censys**—CT logs and Censys together cover >98% of our zone scan results.

In order to understand if this discrepancy applies to commonly visited sites, we can limit the scope of our comparison to certificates discovered through Alexa Top Million scanning, as shown in Table 5.7. Even for these popular sites, IP-based scanning misses 27% of certificates and 65% of sites, due to a lack of SNI—a massive difference compared to the 0.7% that Durumeric et al. found in 2013 [58]. Notably, scans of CT and IPv4 combined provide 98.5% of the certificates presented in our Alexa Top Million scan.

5.4.2 Limitations of Certificate Transparency

While Certificate Transparency is large, and eventual requirement by major browsers seem to make it an authoritative source for certificates on the Internet, this is not the case. We find that there are significant short-comings in its coverage. In particular, we find that non-web use of TLS is missing in significant quantity and that some certificate authorities are slow to adopt Certificate Transparency.

5.4.2.1 Web Focused

We also find that CT is skewed towards web content and away from other TLS-based services, such as webmail, that might not be linked to by websites Google crawls. For example, we find that CT misses 20.8% of certificates with the subdomain `mail` and 50.8% of certificates with a subdomain containing `vpn`. In contrast, CT only misses 4.3% of certificates with the subdomain `www`.

5.4.2.2 CA Bias Participation

While CT is by far the largest perspective, it still misses 9.5% of the certificate universe, including 29.6% of GoDaddy certificates and 28.4% of cPanel certificates, as shown in Table 5.4. None of the CAs in the table submit domain validated (DV) certificates to public logs. In contrast, CT captures 99.3% of CloudFlare certificates and 100% of Let's Encrypt certificates.

5.5 Passive Traffic Monitoring

Passive measurement has many properties that make it different than the other methods used in this chapter. It is the most realistic representation of the Internet's certificates as percieved by users because it is driven by users. However, this view is also less complete than other active methods. This is because all certificates we observe must be requested by a user in our study sample during our measurement period.

5.5.1 Limitations of Scope

The ICSI Notary is the largest passive measurement infrastructure in use for collecting TLS behavior in open publications. It analyzed the traffic of 180,000 users at the time of our study, and over our one month of measurement (longer than any active method) it observed 2.2 billion TLS connections. In spite of this the ICSI Notary was the smallest dataset of certificates collected. A full month of collection was required to very nearly achieve a single Alexa-based scan, although it did produce 3,805 unique certificates. This indicates significant limitations of passive measurement with respect to providing a view into a significant fraction of the Internet and limitations of active

measurement in providing insight into user experience online.

5.5.2 User-driven

The ICSI Notary perspective is derived from passive monitoring of network traffic. It only includes certificates actually seen on the wire, and therefore differs significantly from our other perspectives.

In contrast to our active scans, passive monitoring contains certificates from IPv6 connections. We encountered 822,338 server IP addresses in our Notary dataset. Of these, 8.2% (67,725) were IPv6 addresses, comprising 13% of the observed connections. There were 4,512 certificates that were only encountered on IPv6 addresses, but only 218 of them were not observed in any other perspective. This suggests that IPv6 does not impact conclusions drawn from scanning significantly, but as our passive dataset is relatively restricted, further measurement is necessary to verify this claim. Underscoring the importance of SNI discussed in Section 5.4.1.1, only 9.7% of connections in the Notary dataset did not use SNI. In total, we saw 3,246,725 unique SNI values.

The Notary saw only 3,805 certificates that were not observed by any other perspective. We believe these are due to certificate changes during the longer passive measurement interval. This is supported by the fact that only 34% of the certificates were encountered at all during the last week of the measurement interval. Furthermore, 75% were issued by CloudFlare, which rotates certificates quickly.

There were 68,700 certificates seen by our passive measurement that were not present in Censys. They have a similar composition to the certificates in CT logs that Censys missed. 39% are issued by Let's Encrypt, which requires sites to frequently re-issue certificates. 20% are attributable to Wordpress-hosted blogs and 13% to CloudFlare, services that heavily depend on SNI and would therefore not be seen by IPv4 scans.

5.6 Merger of Certificate Transparency and Censys

As Table 5.8 shows, combining data from CT logs and Censys yields 99.4% coverage of all certificates observed by any of our perspectives and 99.7% of all FQDNs and all sites. However,

since these perspectives are also our two largest data sources, this statistic may be artificially inflated.

Fortunately, scanning all domains in the `.com`, `.net`, and `.org` zone file provides us with ground truth for all sites being served on the root and `www` subdomain in those zones. We can use this to understand what coverage each perspective gives in these zones. While the zones do not contain all subdomains or even all domain names on the Internet, they are a large subset: 153 million unique domains. We compare Censys data and CT logs over the certificates obtained from the zone scans in Table 5.9.

Of these certificates, we find 98.5% are obtained through either Censys or CT logs. Since this is smaller than the corresponding percentage for all certificates we observe, there are likely certificates being hosted in other zones and in other subdomains not observed by any method we use. Therefore, the coverage of IPv4 scans and CT logs on the entire population of certificates on the Internet is overestimated by Table 5.8.

Conversely, the coverage of Censys scanning on the zone dataset is increased to 54%, from 38% over all certificates observed. This is potentially due to certificates in CT logs that are not actively hosted on the Internet (e.g., intranet sites). As a result, the Censys coverage of the entire population of certificates on the Internet is underestimated by Table 5.8.

5.6.1 Benefits to System Administrators

This merger makes it easier for system administrators to keep up to date with the full set of certificates issued for their domains. If the administrator were previously monitoring Certificate Transparency logs, they are now also monitoring those accessed via IPv4 scanning without any change in method. Similarly, if system administrators can now track certificates from both of our largest datasets through free search tools, such as those provided by Censys and `crt.sh`. This allows end users to more reasonably trust that a valid certificate they receive is not misissued.

5.6.2 Load-testing Certificate Transparency

During our initial upload of Censys certificates to Google's certificate transparency logs, we proceeded with a single core uploading certificates as fast as the log servers would accept them.

This, incidentally, was near the limit of certificates per day that the production log servers were able to handle while maintaining service guarantees of maximum merge delay (MMD). This led to the discovery that the production CT logs were not behind a rate limiting HTTP proxy, nor did they perform any rate limiting of their own.

As a result, our initial upload caused the Aviator log to exceed its MMD. This violation of the Certificate Transparancy service guarantee left it untrusted by browsers, and it therefore ceased operation.

This accidental discovery caused Google to ensure the remaining Certificate Transparency logs were placed behind rate limiting proxies with rates set low enough that the MMD can be met. Additionally, more logs have been spun up. As a result, the Certificate Transparency ecosystem is more robust as a result of our modest load test.

5.7 Related Work

This work was inspired by a large body of research focusing on the HTTPS ecosystem and supporting PKI [25, 53, 55, 57, 58, 65, 191, 206, 218]. These works have run the gamut of HTTPS measurement, ranging from the certificate authority ecosystem [58, 65] to cryptographic keys generated without entropy [94], and how operators react to vulnerabilities [57].

In 2010, the EFF launched the SSL Observatory [65], in which they performed a scan of the IPv4 address space over a three month period in order to identify trusted certificate authorities. Later, in 2011, Vratonjic et al. [206] crawled the Alexa Top Million finding that only 5.7% of websites correctly deploy HTTPS. The same year, Holz et al. [100] carried out a similar study, combining active measurements of the Top Million from from several vantage points, with data from passive measurement at a large research institution. They briefly compared the differences of their vantage points, concentrating on differences caused by scan origins.

In 2013, Durumeric et al. [58] analyzed the state of the HTTPS PKI by repeatedly scanning the IPv4 address space. They briefly considered how much of the Alexa Top Million was only accessible using SNI, finding that, at that time, SNI was not widely required. Other studies use

data based on websites on the Top Million [104], and scans of the IPv4 ecosystem [57, 89, 94]. Studies have also been based on combinations of Alexa sites, random domains, known phishing domains data from passive network monitoring [8, 12, 57, 144, 196] Most recently, studies have also investigated TLS deployment outside of HTTPS [56, 99].

Our work does not focus on the questions asked by these individual studies, but instead focuses on *how* these types of studies should be measuring the HTTPS ecosystem. We hope that by better validating different methodologies for studying the HTTPS PKI, we can help future papers obtain more accurate measurements.

5.8 Conclusion

Over the past five years, dozens of studies have measured the HTTPS ecosystem and supporting PKI. Unfortunately, without a clear ground truth, these studies pieced together a view built on a series of fractured and imperfect methodologies. In this work, we investigated these methodologies, finding that IPv4 enumeration no longer provides a representative view of how TLS servers are configured, due to SNI deployment. IPv4 scans miss more than two-thirds of valid certificates and associated measurements can differ dramatically from site-based approaches. Certificate Transparency provides a new perspective, which finds 90.5% of certificates observed in this study, but is skewed towards a few authorities that submit the certificates they issue. We find that a more comprehensive yet readily accessible methodology is to use a combination of CT and Censys data, which together account for 99.4% of our observed certificates. To this end, we are working with the Censys team to implement continuous certificate synchronization between Censys and Google's CT logs, which will soon make either data source a nearly comprehensive view of trusted HTTPS certificates.

CHAPTER VI

Conclusion and Future Work

This dissertation has explored three separate software systems with security properties on the Internet: online censorship, web tracking, and TLS PKI. We analyzed each of these with Internet measurement to give the end users transparency into censorship, privacy from trackers, and trust in PKI. This has led to a more equitable Internet in several ways. Half of a million users have Refraction Networking clients in their hands as a way to circumvent censorship. Browser policies that affect Kazakhstanis' access to the Internet were shaped by application-layer remote censorship measurement by the Censored Planet project. Recommendations of which browser extension to use to defend user privacy can now be shaped by direct evidence. Each of these are examples of how transparency, trust, and privacy can be enhanced with Internet Measurement.

Internet measurement provides insights that are not otherwise attainable, and in this dissertation I attribute this to four capabilities of Internet measurement. First, Internet measurement enables discovery of hosts of a particular character for use, such as echo servers or filtering potential decoys. Second, it enables insight into the topology of the network infrastructure, as used in identifying initial potential decoys. Third, it enables measurement of populations and their behaviors on the Internet, such as websites and their use of third party resources. Finally, it enables identification or measurement of emergent properties, such as privacy impacts due to the common third parties online and biases introduced to TLS measurement through the SNI extension.

Throughout this dissertation, we have applied these four capabilities to a limited scope of

problems users face. However, there are far more problems users face online than we address here; challenges to an equitable Internet are not limited to those in this dissertation, and seem to be creeping in scope. In the remainder of this thesis, I will identify some ways I see potential in the application of these properties of Internet measurement to improving the Internet for end users.

One direction of future research is study of the quality of Internet access. Prior work [141] has demonstrated that from a global, site-level perspective, the Internet is not equally accessible. This leads to many questions that confront more subtle aspects of this phenomenon. Can we observe different websites blocking beyond just the homepage, such as not permitting checkout when a billing address is in a sanctioned country? Can we observe blocking based on geographic location between different parts of a single country? Moreover, this work elides the question of quality of Internet connection. In particular, within a single country what are the differences between under-served and adequately served regions in Internet quality and how does this affect quality of service for essential service page loads? If Internet quality is quantified in terms of up-time, latency, and bandwidth, these are population properties that Internet measurement could be applied to measure.

Another direction of future research is to more accurately model users in the study of online tracking. And improvement we made over prior work was to weight websites in a Zipfian manner to approximate site visit frequency. However, this leaves many aspects of user experience omitted. Can we integrate work that reasonably automates page traversal in a way that resembles user behavior to a browser instrumented for tracking detection? Does this impact how trackers treat a user? Additionally, our study occurred from a subnet known to perform scanning and does not typically have users. Does tracking behavior become more representative of user experience when performed in typical user network locations? Moreover, does your network location create a baseline belief in the advertiser models about your interests or value in advertising?

The final question of each of the last two proposed research directions can be applied together to gain deep insight into discrimination via the Internet. Consider a case study in the United States where access to the Internet is not a given and targeted advertisements have been found

to be discriminatory. These questions, answered quantitatively at a national level, could provide significant insight into emergent properties and help measure our progress in correcting them.

An additional direction of future research in privacy is to focus on the network level. Prior work in enhancing Tor circuit selection algorithms [63, 173] has used topology to quantify effectiveness of privacy protections. Understanding a comprehensive baseline using a more accurate user model, topology information, DNS leaks in recursive resolution, and centralized web hosting may be useful in the development of weaker, but more usable privacy enhancing technologies. At very least these would help users understand the degree of exposure they experience on the Internet today.

In our work on Refraction Networking, we identified that a major impediment to dial speed was the frequency of stations not being on path between a given client and decoy. An avenue of improvement for Refraction Networking performance is therefore performing router-level prediction of routes to reduce the dial failure frequency. This is a challenge, not only in accurately predicting route choice at a fine-grained level, but also in performing this for each client without obtaining the client IP address. Two potential solutions for the latter problem are immediately apparent, with trade-offs: developing a client IP address indexed prefix tree of decoys and shipping some data about routing to each client to allow them to make the decision. The trade-off here is a question of how little data we can send to the client (reducing overhead), how accurate we can be (improving performance), and how much computation we force the client to do (improving performance). This is to say nothing of the challenges in developing fine-grained route prediction when doing so at the AS level is still being improved [185].

A direction of future research in Internet measurement that is not closely related to any chapter of this dissertation is the study of channels of radicalization. There is some prior work in understanding how information travels through fringe web communities [136, 216], state sponsored behavior [217], and the direction of users within fringe communities on YouTube [172]. However this area has many open questions. Some may be answered in part with Internet measurement, such as host discovery and web crawls. How do platforms less centralized than YouTube direct users between communities? Mastodon has been studied from a structural level [169], but the content therein and

how users move within communities has not seen similar focus.

Additionally, there are some directions of future research that may not require Internet measurement, but work in this dissertation points to.

The first is improvement in scalability of Refraction Networking. Conjure [80] already takes significant steps toward doing this, removing the hung open connection to the decoy, reducing the burden of hosting a refraction station. Future work could deploy Conjure for practical use or identify improvements that enable it for deployment at a Tier-1 ISP. A Tier-1 ISP deployment would cement the utility of Refraction Networking, due to the high number of decoys available. Additionally, future work making Conjure resistant to traffic analysis, to the standard of Slitheen [26], would be compelling; Refraction Networking fits in as a strong back-up circumvention technique.

The second is focused study of ad exchange behavior. In particular, precisely what information they leak to bidders, what techniques they use to fingerprint users, and how they handle opt-out behaviors. We now know that a few dozen ad exchanges are responsible for the bulk of out-of-band information sharing in the web tracking ecosystem. This is a small enough number that they can be analyzed manually, and better understand the major impact they have and how to better defend against them. Additionally, constant monitoring of the important findings can be put in place with measurement techniques that prevent unknowns from growing underneath our knowledge.

This dissertation has been guided by improvements to users' transparency, trust, and privacy, particularly with Internet measurement. While Internet measurement is a powerful tool to understand systems, it is not a panacea. In particular, work of technical discovery must be matched by work to deploy and teach the lessons learned in order to improve privacy and transparency. Thus, future work should consider how users are brought into the loop in its developments and lessons. We should take deliberate steps to ensure that people are benefiting from what we do.

BIBLIOGRAPHY

All URLs available on The Wayback Machine at April 22, 2020 or earlier.

[1] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juarez, Arvind Narayanan, and Claudia Diaz. The web never forgets: Persistent tracking mechanisms in the wild. In *ACM Conference on Computer and Communications Security (CCS)*, 2014.

[2] Gunes Acar, Marc Juarez, Nick Nikiforakis, Claudia Diaz, Seda Gürses, Frank Piessens, and Bart Preneel. FPDetective: dusting the web for fingerprinters. In *ACM Conference on Computer and Communications Security (CCS)*, 2013.

[3] AdGuard. How to create your own ad filters. Available: https://kb.adguard.com/en/general/how-to-create-your-own-ad-filters.

[4] AdGuard. Search ads and self-promotion. Available: https://kb.adguard.com/en/general/search-ads-and-self-promotion.

[5] AdGuard. The world's most advanced ad blocker! Available: https://adguard.com/en/welcome.html.

[6] Sadia Afroz and David Fifield. Timeline of Tor censorship, 2007. Available: http://www1.icsi.berkeley.edu/~sadia/tor_timeline.pdf.

[7] Lalit Agarwal, Nisheeth Shrivastava, Sharad Jaiswal, and Saurabh Panjwani. Do not embarrass: re-examining user concerns for online tracking and advertising. In *USENIX Symposium on Usable Security and Privacy (SOUPS)*, 2013.

[8] Devdatta Akhawe, Johanna Amann, Matthias Vallentin, and Robin Sommer. Here's my cert, so trust me, maybe? Understanding TLS errors on the web. In *International Conference on World Wide Web (WWW)*, 2013.

[9] Alexa Internet, Inc. Alexa Top 1,000,000 Sites. Available: http://s3.amazonaws.com/alexa-static/top-1m.csv.zip.

[10] Alphabet. Alphabet announces first quarter 2019 results, April 2019. Available: https://abc.xyz/investor/static/pdf/2019Q1_alphabet_earnings_release.pdf.

[11] Johanna Amann, Oliver Gasser, Quirin Scheitle, Lexi Brent, Georg Carle, and Ralph Holz. Mission accomplished? https security after diginotar. In *ACM Internet Measurement Conference (IMC)*, 2017.

[12] Johanna Amann, Robin Sommer, Matthias Vallentin, and Seth Hall. No attack necessary: The surprising dynamics of SSL trust relationships. In *ACM Computer Security Applications Conference (CSAC)*, 2013.

[13] Johanna Amann, Matthias Vallentin, Seth Hall, and Robin Sommer. Extracting certificates from live traffic: A near real-time SSL notary service, November 2012. Technical Report, ICSI TR-12-014.

[14] Anonymous. Towards a comprehensive picture of the Great Firewall's DNS censorship. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2014.

[15] Simurgh Aryan, Homa Aryan, and J Alex Halderman. Internet censorship in Iran: A first look. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2013.

[16] Hadi Asghari, Michel Van Eeten, and Milton Mueller. Unraveling the economic and political drivers of deep packet inspection. In *GigaNet Annual Symposium*, 2012.

[17] Michael Backes, Aniket Kate, Matteo Maffei, and Kim Pecina. Obliviad: Provably secure and practical online behavioral advertising. In *IEEE Symposium on Security and Privacy (Oakland)*, 2012.

[18] Ricardo A Baeza-Yates and Gaston H Gonnet. Fast text searching for regular expressions or automaton searching on tries. *Journal of the ACM (JACM)*, 43(6), 1996.

[19] Rebecca Balebako, Pedro Leon, Richard Shay, Blase Ur, Yang Wang, and L Cranor. Measur-

ing the effectiveness of privacy tools for limiting behavioral advertising. In *IEEE Workshop on Web 2.0 Security and Privacy (W2SP)*, 2012.

[20] Paul Barford, Igor Canadi, Darja Krushevskaja, Qiang Ma, and S Muthukrishnan. Adscape: Harvesting and analyzing online display ads. In *International Conference on World Wide Web (WWW)*, 2014.

[21] Muhammad Ahmad Bashir, Sajjad Arshad, William Robertson, and Christo Wilson. Tracing information flows between ad exchanges using retargeted ads. In *USENIX Security Symposium*, 2016.

[22] Muhammad Ahmad Bashir, Umar Farooq, Maryam Shahid, Muhammad Fareed Zaffar, and Christo Wilson. Quantity vs. quality: Evaluating user interest profiles using ad preference managers. In *Internet Society Network and Distributed System Security Symposium (NDSS)*, 2019.

[23] Muhammad Ahmad Bashir and Christo Wilson. Diffusion of user tracking data in the online advertising ecosystem. *Proceedings on Privacy Enhancing Technologies (PoPET)*, 2018(4), 2018.

[24] Daniel J Bernstein, Mike Hamburg, Anna Krasnova, and Tanja Lange. Elligator: Elliptic-curve points indistinguishable from uniform random strings. In *ACM Conference on Computer and Communications Security (CCS)*, 2013.

[25] Benjamin Beurdouche, Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Markulf Kohlweiss, Alfredo Pironti, Pierre-Yves Strub, and Jean Karim Zinzindohoue. A messy state of the union: Taming the composite state machines of TLS. In *IEEE Symposium on Security and Privacy (Oakland)*, 2015.

[26] Cecylia Bocovich and Ian Goldberg. Slitheen: Perfectly imitated decoy routing through traffic replacement. In *ACM Conference on Computer and Communications Security (CCS)*, 2016.

[27] Cecylia Bocovich and Ian Goldberg. Secure asymmetry and deployability for decoy routing systems. *Proceedings on Privacy Enhancing Technologies*, 2018(3), 2018.

[28] Katrina Booker. "i was devastated": Tim Berners-Lee, the man who created the World Wide Web, has some regrets, July 2018. Available: https://www.vanityfair.com/news/2018/07/the-man-who-created-the-world-wide-web-has-some-regrets.

[29] Justin Brookman, Phoebe Rouge, Aaron Alva, and Christina Yeung. Cross-device tracking: Measurement and disclosures. *Proceedings on Privacy Enhancing Technologies (PoPET)*, 2017(2), 2017.

[30] Sam Burnett and Nick Feamster. Encore: Lightweight measurement of web censorship with cross-origin requests. In *ACM SIG on Data Communication Conference (SIGCOMM)*, 2015.

[31] Aaron Cahn, Scott Alfeld, Paul Barford, and S Muthukrishnan. An empirical study of web cookies. In *International Conference on World Wide Web (WWW)*, 2015.

[32] Juan Miguel Carrascosa, Jakub Mikians, Ruben Cuevas, Vijay Erramilli, and Nikolaos Laoutaris. I always feel like somebody's watching me: measuring online behavioural advertising. In *ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, 2015.

[33] Claude Castelluccia, Mohamed-Ali Kaafar, and Minh-Dung Tran. Betrayed by your ads! In *Proceedings of the Privacy Enhancing Technologies Symposium (PETS)*, 2012.

[34] Claude Castelluccia, Lukasz Olejnik, and Tran Minh-Dung. Selling off privacy at auction. In *Internet Society Network and Distributed System Security Symposium (NDSS)*, 2014.

[35] Censored Planet. Investigating security and privacy violations on the Internet, March 2020. Available: https://censoredplanet.org.

[36] Certificate Transparency: Extended validation in Chrome. Available: https://www.certificate-transparency.org/ev-ct-plan.

[37] Jacopo Cesareo, Josh Karlin, Michael Schapira, and Jennifer Rexford. Optimizing the placement of implicit proxies, June 2012. Technical Report, Available: http://www.cs.princeton.edu/~jrex/papers/decoy-routing.pdf.

[38] Abdelberi Chaabane, Terence Chen, Mathieu Cunche, Emiliano De Cristofaro, Arik Fried-

man, and Mohamed Ali Kaafar. Censorship in the wild: Analyzing Internet filtering in Syria. In *ACM Internet Measurement Conference (IMC)*, 2014.

[39] Farah Chanchary and Sonia Chiasson. User perceptions of sharing, advertising, and tracking. In *USENIX Symposium on Usable Security and Privacy (SOUPS)*, 2015.

[40] Citizen Lab. Block test list. Available: https://github.com/citizenlab/test-lists.

[41] Richard Clayton, Steven J. Murdoch, and Robert N. M. Watson. Ignoring the Great Firewall of China. In *Proceedings of the Privacy Enhancing Technologies Symposium (PETS)*, 2006.

[42] Cliqz. Faster, safer, smarter browsing. Available: https://www.ghostery.com/.

[43] Common Crawl. Available: https://commoncrawl.org/.

[44] Jedidiah R Crandall, Masashi Crete-Nishihata, Jeffrey Knockel, Sarah McKune, Adam Senft, Diana Tseng, and Greg Wiseman. Chat program censorship and surveillance in China: Tracking TOM-Skype and Sina UC. *First Monday*, 18(7), 2013.

[45] Jedidiah R Crandall, Daniel Zinn, Michael Byrd, Earl T Barr, and Rich East. ConceptDoppler: A weather tracker for Internet censorship. In *ACM Conference on Computer and Communications Security (CCS)*, 2007.

[46] Jakub Dalek, Bennett Haselton, Helmi Noman, Adam Senft, Masashi Crete-Nishihata, Phillipa Gill, and Ronald J. Deibert. A method for identifying and confirming the use of URL filtering products for censorship. In *ACM Internet Measurement Conference (IMC)*, 2013.

[47] Martin Degeling, Christine Utz, Christopher Lentzsch, Henry Hosseini, Florian Schaub, and Thorsten Holz. We value your privacy... now take some cookies: Measuring the GDPR's impact on web privacy, November 2018. Technical Report, arXiv preprint arXiv:1808.05096.

[48] Disconnect. Tracking protection lists. Available: https://disconnect.me/trackerprotection.

[49] Disconnect. We protect your privacy. Available: https://disconnect.me/.

[50] David Dittrich and Erin Kenneally. The Menlo Report: Ethical principles guiding information and communication technology research. Technical report, U.S. Department of Homeland Security, 2012.

[51] Lucas Dixon, Thomas Ristenpart, and Thomas Shrimpton. Network traffic obfuscation and

automated Internet censorship. In *IEEE Symposium on Security and Privacy (Oakland)*, 2016.

[52] Claire Dolin, Ben Weinshel, Shawn Shan, Chang Min Hahn, Euirim Choi, Michelle L Mazurek, and Blase Ur. Unpacking perceptions of data-driven inferences underlying online targeting and personalization. In *ACM SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 2018.

[53] The DROWN attack. Available: https://drownattack.com/.

[54] DuckDuckGo. Duckduckgo privacy essentials. Available: https://addons.mozilla.org/en-US/firefox/addon/duckduckgo-for-firefox/.

[55] Zakir Durumeric, David Adrian, Ariana Mirian, Michael Bailey, and J. Alex Halderman. Censys: A search engine backed by Internet-wide scanning. In *ACM Conference on Computer and Communications Security (CCS)*, 2015.

[56] Zakir Durumeric, David Adrian, Ariana Mirian, James Kasten, Elie Bursztein, Nicolas Lidzborski, Kurt Thomas, Vijay Eranti, Michael Bailey, and J Alex Halderman. Neither snow nor rain nor MITM... an empirical analysis of email delivery security. In *ACM Internet Measurement Conference (IMC)*, 2015.

[57] Zakir Durumeric, James Kasten, David Adrian, J. Alex Halderman, Michael Bailey, Frank Li, Nicolas Weaver, Johanna Amann, Jethro Beekman, Mathias Payer, and Vern Paxson. The matter of Heartbleed. In *ACM Internet Measurement Conference (IMC)*, 2014.

[58] Zakir Durumeric, James Kasten, Michael Bailey, and J. Alex Halderman. Analysis of the HTTPS certificate ecosystem. In *ACM Internet Measurement Conference (IMC)*, 2013.

[59] Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. ZMap: Fast Internet-wide scanning and its security applications. In *USENIX Security Symposium*, 2013.

[60] Madeline Earp. Kazakhstan's move to control internet prompts censorship, surveillance concerns, July 2019. Available: https://cpj.org/blog/2019/07/kazakhstans-move-to-control-internet-prompts-censo.php.

[61] Donald Eastlake 3rd. Transport layer security (TLS) extensions: Extension definitions. RFC 6066, 2011.

[62] Peter Eckersley. How unique is your web browser? In *Proceedings of the Privacy Enhancing Technologies Symposium (PETS)*, 2010.

[63] Matthew Edman and Paul Syverson. As-awareness in Tor path selection. In *ACM Conference on Computer and Communications Security (CCS)*, pages 380–389.

[64] Elastic, Co. Elastic stack and product documentation. Available: https://www.elastic.co/guide/index.html.

[65] Electronic Frontier Foundation. The EFF SSL observatory. Available: https://www.eff.org/observatory.

[66] Daniel Ellard, Alden Jackson, Christine Jones, Victoria Manfredi, W. Timothy Strayer, Bishal Thapa, and Megan Van Welie. Rebound: Decoy routing on asymmetric routes via error messages. In *IEEE Conference on Local Computer Networks (LCN)*, 2015.

[67] Steven Englehardt and Arvind Narayanan. Online tracking: A 1-million-site measurement and analysis. In *ACM Conference on Computer and Communications Security (CCS)*, 2016.

[68] Steven Englehardt, Dillon Reisman, Christian Eubank, Peter Zimmerman, Jonathan Mayer, Arvind Narayanan, and Edward W Felten. Cookies that give you away: The surveillance implications of web tracking. In *International Conference on World Wide Web (WWW)*, 2015.

[69] Roya Ensafi, Jeffrey Knockel, Geoffrey Alexander, and Jedidiah R Crandall. Detecting intentional packet drops on the Internet via TCP/IP side channels. In *Passive and Active Measurement Conference (PAM)*, 2014.

[70] Roya Ensafi, Philipp Winter, Abdullah Mueen, and Jedidiah R Crandall. Analyzing the Great Firewall of China over space and time. *Proceedings on Privacy Enhancing Technologies (PoPET)*, 2015.

[71] Eyeo GmbH. Allowing acceptable ads in Adblock Plus. Available: https://adblockplus.org/acceptable-ads.

[72] Eyeo GmbH. Homepage. Available: https://adblockplus.org/.

[73] Eyeo GmbH. Writing AdBlock Plus Filters. Available: https://adblockplus.org/filters.

[74] Marjan Falahrastegar, Hamed Haddadi, Steve Uhlig, and Richard Mortier. Tracking personal identifiers across the web. In *Passive and Active Measurement Conference (PAM)*, 2016.

[75] Arturo Filastò and Jacob Appelbaum. OONI: Open Observatory of Network Interference. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2012.

[76] FortiNet. FortiGuard labs web filter. Available: https://fortiguard.com/webfilter.

[77] Matthew Fredrikson and Benjamin Livshits. Repriv: Re-imagining content personalization and in-browser privacy. In *IEEE Symposium on Security and Privacy (Oakland)*, 2011.

[78] Freedom House. Freedom on the net 2016, November 2016.

[79] Sergey Frolov, Fred Douglas, Will Scott, Allison McDonald, Benjamin VanderSloot, Rod Hynes, Adam Kruger, Michalis Kallitsis, David G Robinson, Steve Schultze, et al. An isp-scale deployment of tapdance. In *7th USENIX Workshop on Free and Open Communications on the Internet (FOCI 17)*, 2017.

[80] Sergey Frolov, Jack Wampler, Sze Chuen Tan, J. Alex Halderman, Nikita Borisov, and Eric Wustrow. Conjure: Summoning proxies from unused address space. In *ACM Conference on Computer and Communications Security (CCS)*, 2019.

[81] King-wa Fu, Chung-hong Chan, and Michael Chau. Assessing censorship on microblogs in China: Discriminatory keyword analysis and the real-name registration policy. *IEEE Internet Computing*, 17(3), 2013.

[82] Genevieve Gebhart and Tadayoshi Kohno. Internet censorship in Thailand: User practices and potential threats. In *IEEE Symposium on Security and Privacy (EuroS&P)*, 2017.

[83] Arthur Gervais, Alexandros Filios, Vincent Lenders, and Srdjan Capkun. Quantifying web adblocker privacy. In *European Symposium on Research in Computer Security (ESORICS)*, 2017.

[84] Arpita Ghosh, Mohammad Mahdian, R Preston McAfee, and Sergei Vassilvitskii. To match or not to match: Economics of cookie matching in online advertising. *ACM Transactions on Economics and Computation (TEAC)*, 3(2), 2015.

[85] Phillipa Gill, Vijay Erramilli, Augustin Chaintreau, Bala Krishnamurthy, Dina Papagiannaki, and Pablo Rodriguez. Follow the money: Understanding economics of online aggregation and advertising. In *ACM Internet Measurement Conference (IMC)*, 2013.

[86] Richard Gomer, Eduarda Mendes Rodrigues, Natasa Milic-Frayling, and MC Schraefel. Network analysis of third party tracking: User exposure to tracking cookies through search. In *IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (WI)*, 2013.

[87] Devashish Gosain, Anshika Agarwal, Sambuddho Chakravarty, and HB Acharya. The devil's in the details: Placing decoy routers in the internet. In *Annual Computer Security Applications Conference (ACSAC)*, 2017.

[88] Grafana Labs. Grafana documentation. Available: https://grafana.com/docs/.

[89] Xiaojing Gu and Xingsheng Gu. On the detection of fake certificates via attribute correlation. *Entropy*, 2015, 2015.

[90] Dimitar Gueorguiev, Li Shao, and Charles Crabtree. Blurring the lines: Rethinking censorship under autocracy, 2017.

[91] Saikat Guha, Bin Cheng, and Paul Francis. Challenges in measuring online advertising systems. In *ACM Internet Measurement Conference (IMC)*, 2010.

[92] Saikat Guha, Bin Cheng, and Paul Francis. Privad: Practical privacy in online advertising. In *USENIX Conference on Networked Systems Design and Implementation (NSDI)*, 2011.

[93] Miguel Helft. Marketers can glean private data on Facebook. New York Times, October 2010. Available: https://www.nytimes.com/2010/10/23/technology/23facebook.html.

[94] Nadia Heninger, Zakir Durumeric, Eric Wustrow, and J. Alex Halderman. Mining your Ps and Qs: Detection of widespread weak keys in network devices. In *USENIX Security Symposium*, 2012.

[95] Kashmir Hill. How Target figured out a teen girl was pregnant before her father did. Forbes, February 2012. Available: https://www.forbes.com/sites/kashmirhill/2012/02/16/how-target-figured-out-a-teen-girl-was-pregnant-before-her-father-did.

[96] Raymond Hill. Static filter syntax. Available: https://github.com/gorhill/uBlock/wiki/Static-filter-syntax.

[97] Raymond Hill. uBlock. Available: https://github.com/gorhill/uBlock.

[98] Pieter Hintjens. *ZeroMQ: messaging for many applications*. " O'Reilly Media, Inc.".

[99] Ralph Holz, Johanna Amann, Oliver Mehani, Matthias Wachs, and Mohamed Ali Kaafar. TLS in the wild: An Internet-wide analysis of TLS-based protocols for electronic communication. In *Internet Society Network and Distributed System Security Symposium (NDSS)*, 2016.

[100] Ralph Holz, Lothar Braun, Nils Kammenhuber, and Georg Carle. The SSL landscape: A thorough analysis of the x.509 PKI using active and passive measurements. In *ACM Internet Measurement Conference (IMC)*, 2011.

[101] Amir Houmansadr, Giang T. K. Nguyen, Matthew Caesar, and Nikita Borisov. Cirripede: Circumvention infrastructure using router redirection with plausible deniability. In *ACM Conference on Computer and Communications Security (CCS)*, 2011.

[102] Amir Houmansadr, Giang TK Nguyen, Matthew Caesar, and Nikita Borisov. Cirripede: Circumvention infrastructure using router redirection with plausible deniability. In *ACM Conference on Computer and Communications Security (CCS)*, 2011.

[103] Amir Houmansadr, Edmund L Wong, and Vitaly Shmatikov. No direction home: The true cost of routing around decoys. In *Internet Society Network and Distributed System Security Symposium (NDSS)*, 2014.

[104] Lin-Shung Huang, Shrikant Adhikarla, Dan Boneh, and Collin Jackson. An experimental study of TLS forward secrecy deployments. *IEEE Internet Computing*, 18(6), 2014.

[105] Muhammad Ikram, Hassan Jameel Asghar, Mohamed Ali Kaafar, Anirban Mahanti, and Balachandar Krishnamurthy. Towards seamless tracking-free web: Improved detection of trackers via one-class learning. *Proceedings on Privacy Enhancing Technologies (PoPET)*, 2017(1), 2017.

[106] Internet Archive. Wayback machine. Available: https://archive.org/.

[107] Umar Iqbal, Zubair Shafiq, and Zhiyun Qian. The ad wars: retrospective measurement and analysis of anti-adblock filter lists. In *ACM Internet Measurement Conference (IMC)*, 2017.

[108] Umar Iqbal, Zubair Shafiq, Peter Snyder, Shitong Zhu, Zhiyun Qian, and Benjamin Livshits. Adgraph: A machine learning approach to automatic and effective adblocking. In *IEEE Symposium on Security and Privacy (Oakland)*, 2020.

[109] Collin Jackson, Andrew Bortz, Dan Boneh, and John C Mitchell. Protecting browser state from web privacy attacks. In *International Conference on World Wide Web (WWW)*, 2006.

[110] Dongseok Jang, Ranjit Jhala, Sorin Lerner, and Hovav Shacham. An empirical study of privacy-violating information flows in JavaScript web applications. In *ACM Conference on Computer and Communications Security (CCS)*, 2010.

[111] Carlos Jensen, Chandan Sarkar, Christian Jensen, and Colin Potts. Tracking website data-collection and privacy practices with the iWatch web crawler. In *USENIX Symposium on Usable Security and Privacy (SOUPS)*, 2007.

[112] Ben Jones, Roya Ensafi, Nick Feamster, Vern Paxson, and Nick Weaver. Ethical concerns for censorship measurement. In *ACM SIG on Data Communication Conference (SIGCOMM)*, 2015.

[113] Ben Jones, Tzu-Wen Lee, Nick Feamster, and Phillipa Gill. Automated detection and fingerprinting of censorship block pages. In *ACM Internet Measurement Conference (IMC)*, 2014.

[114] J. C. Jones. 124 days of Let's Encrypt, April 2016. Available: https://tacticalsecret.com/124-days-of-lets-encrypt/.

[115] Vasiliki Kalavri, Jeremy Blackburn, Matteo Varvello, and Konstantina Papagiannaki. Like a pack of wolves: Community structure of web trackers. In *Passive and Active Measurement Conference (PAM)*, 2016.

[116] Michael Kan. Russia to block 9 VPNs for rejecting censorship demand. Available: https://www.pcmag.com/news/russia-to-block-9-vpns-for-rejecting-censorship-demand.

[117] Arjaldo Karaj, Sam Macbeth, Rémi Berson, and Josep M Pujol. WhoTracks.Me: Monitoring

the online tracking landscape at scale, October 2018. Technical Report, arXiv preprint arXiv:1804.08959.

[118] Josh Karlin, Daniel Ellard, Alden W. Jackson, Christine E. Jones, Greg Lauer, David P. Mankins, and W. Timothy Strayer. Decoy routing: Toward unblockable Internet communication. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2011.

[119] Jeffrey Kline, Aaron Cahn, and Paul Barford. Placement laundering and the complexities of attribution in online advertising. 2018.

[120] Jeffrey Knockel, Jedidiah R Crandall, and Jared Saia. Three researchers, five conjectures: An empirical analysis of TOM-Skype censorship and surveillance. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2011.

[121] Richie Koch. How to block online trackers, 2018. Available: https://protonvpn.com/blog/block-online-tracking/.

[122] Balachander Krishnamurthy, Delfina Malandrino, and Craig E Wills. Measuring privacy loss and the impact of privacy protection in web browsing. In *USENIX Symposium on Usable Security and Privacy (SOUPS)*, 2007.

[123] Balachander Krishnamurthy and Craig Wills. Privacy diffusion on the Web: a longitudinal perspective. In *International Conference on World Wide Web (WWW)*, 2009.

[124] Ben Laurie, Adam Langley, and E Kasper. Certificate Transparency, 2013. Available: http://www.certificate-transparency.org/.

[125] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. Tranco: A research-oriented top sites ranking hardened against manipulation. In *Internet Society Network and Distributed System Security Symposium (NDSS)*, 2019.

[126] Mathias Lécuyer, Guillaume Ducoffe, Francis Lan, Andrei Papancea, Theofilos Petsios, Riley Spahn, Augustin Chaintreau, and Roxana Geambasu. XRay: Enhancing the web's transparency with differential correlation. In *USENIX Security Symposium*, 2014.

[127] Arjen K. Lenstra, James P. Hughes, Maxime Augier, Joppe W. Bos, Thorsten Kleinjung, and Christophe Wachter. Public keys. In *International Cryptology Conference (CRYPTO)*, 2012.

[128] Pedro Giovanni Leon, Blase Ur, Yang Wang, Manya Sleeper, Rebecca Balebako, Richard Shay, Lujo Bauer, Mihai Christodorescu, and Lorrie Faith Cranor. What matters to users?: factors that affect users' willingness to share information with online advertisers. In *USENIX Symposium on Usable Security and Privacy (SOUPS)*, 2013.

[129] Ada Lerner, Anna Kornfeld Simpson, Tadayoshi Kohno, and Franziska Roesner. Internet Jones and the raiders of the lost trackers: An archaeological study of web tracking from 1996 to 2016. In *USENIX Security Symposium*, 2016.

[130] Let's Encrypt: Certificates. Available: https://letsencrypt.org/certificates/.

[131] Tai-Ching Li, Huy Hang, Michalis Faloutsos, and Petros Efstathopoulos. Trackadvisor: Taking back browsing privacy from third-party trackers. In *Passive and Active Measurement Conference (PAM)*, 2015.

[132] Rebecca MacKinnon. China's censorship 2.0: How companies censor bloggers. *First Monday*, 14(2), 2009.

[133] Miguel Malheiros, Charlene Jennett, Snehalee Patel, Sacha Brostoff, and Martina Angela Sasse. Too close for comfort: A study of the effectiveness and acceptability of rich-media personalized advertising. In *ACM SIGCHI Conference on Human Factors in Computing Systems (CHI)*, 2012.

[134] Matthew Malloy, Mark McNamara, Aaron Cahn, and Paul Barford. Ad blockers: Global prevalence and impact. In *ACM Internet Measurement Conference (IMC)*, 2016.

[135] Victoria Manfredi and Pi Songkuntham. Multiflow: Cross-connection decoy routing using TLS 1.3 session resumption. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2018.

[136] Binny Mathew, Ritam Dutt, Pawan Goyal, and Animesh Mukherjee. Spread of hate speech in online social media. In *ACM Conference on Web Science (WebSci)*, 2019.

[137] Arunesh Mathur, Jessica Vitak, Arvind Narayanan, and Marshini Chetty. Characterizing the use of browser-based blocking extensions to prevent online tracking. In *USENIX Symposium on Usable Security and Privacy (SOUPS)*, 2018.

[138] MaxMind. Available: https://www.maxmind.com/.

[139] Jonathan R Mayer and John C Mitchell. Third-party web tracking: Policy and technology. In *IEEE Symposium on Security and Privacy (Oakland)*, 2012.

[140] Aleecia M McDonald and Lorrie Faith Cranor. Americans' attitudes about Internet behavioral advertising practices. In *ACM Workshop on Privacy in the Electronic Society (WPES)*, 2010.

[141] Allison McDonald, Matthew Bernhard, Luke Valenta, Benjamin VanderSloot, Will Scott, Nick Sullivan, J Alex Halderman, and Roya Ensafi. 403 forbidden: A global view of CDN geoblocking. In *ACM Internet Measurement Conference (IMC)*, 2018.

[142] William Melicher, Mahmood Sharif, Joshua Tan, Lujo Bauer, Mihai Christodorescu, and Pedro Giovanni Leon. (Do Not) Track me sometimes: users' contextual preferences for web tracking. *Proceedings on Privacy Enhancing Technologies (PoPET)*, 2016(2), 2016.

[143] Georg Merzdovnik, Markus Huber, Damjan Buhov, Nick Nikiforakis, Sebastian Neuner, Martin Schmiedecker, and Edgar Weippl. Block me if you can: A large-scale study of tracker-blocking tools. In *IEEE Symposium on Security and Privacy (EuroS&P)*, 2017.

[144] Mishari Al Mishari, Emiliano De Cristofaro, Karim M. El Defrawy, and Gene Tsudik. Harvesting SSL certificate data to mitigate web-fraud. *International Journal of Network Security*, 14(6), 2012.

[145] Mozilla. Content blocking. Available: https://support.mozilla.org/en-US/kb/content-blocking.

[146] Mozilla Foundation. Public suffix list. Available: https://publicsuffix.org/.

[147] Muhammad Haris Mughees, Zhiyun Qian, and Zubair Shafiq. Detecting anti ad-blockers in the wild. *Proceedings on Privacy Enhancing Technologies (PoPET)*, 2017(3), 2017.

[148] Milad Nasr and Amir Houmansadr. Game of decoys: Optimal decoy routing through game theory. In *ACM Conference on Computer and Communications Security (CCS)*, 2016.

[149] Milad Nasr, Hadi Zolfaghari, and Amir Houmansadr. The waterfall of liberty: Decoy routing circumvention that resists routing attacks. In *ACM Conference on Computer and Communications Security (CCS)*, 2017.

[150] Milad Nasr, Hadi Zolfaghari, and Amir Houmansadr. The waterfall of liberty: Decoy routing circumvention that resists routing attacks. In *ACM Conference on Computer and Communications Security (CCS)*, 2017.

[151] National Commission for the Protection of Human Subjects of Biomedical and Behavioral Research. *The Belmont Report: Ethical Principles and Guidelines for the Protection of Human Subjects of Research*. 1978.

[152] Nick Nikiforakis, Wouter Joosen, and Benjamin Livshits. Privaricator: Deceiving fingerprinters with little white lies. In *International Conference on World Wide Web (WWW)*, 2015.

[153] Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *IEEE Symposium on Security and Privacy (Oakland)*, 2013.

[154] Rishab Nithyanand, Sheharbano Khattak, Mobin Javed, Narseo Vallina-Rodriguez, Marjan Falahrastegar, Julia E Powles, Emiliano De Cristofaro, Hamed Haddadi, and Steven J Murdoch. Adblocking and counter blocking: A slice of the arms race. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2016.

[155] Ntop. PF_RING. Available: http://www.ntop.org/products/pf_ring.

[156] OpenNet Initiative. Jordan, August 2009. Available: https://opennet.net/research/profiles/jordan.

[157] OpenNet Initiative. South Korea, August 2012. Available: https://opennet.net/research/profiles/south-korea.

[158] Panagiotis Papadopoulos, Nicolas Kourtellis, and Evangelos P Markatos. Cookie synchronization: Everything you always wanted to know but were afraid to ask. In *Passive and Active Measurement Conference (PAM)*, 2019.

[159] Panagiotis Papadopoulos, Nicolas Kourtellis, Pablo Rodriguez Rodriguez, and Nikolaos Laoutaris. If you are not paying for it, you are the product: How much do advertisers pay to reach you? In *ACM Internet Measurement Conference (IMC)*, 2017.

[160] Craig Partridge and Mark Allman. Addressing ethical considerations in network measurement papers. In *Workshop on Ethics in Networked Systems Research (NS Ethics@SIGCOMM)*, 2015.

[161] Paul Pearce, Roya Ensafi, Frank Li, Nick Feamster, and Vern Paxson. Augur: Internet-wide detection of connectivity disruptions. In *IEEE Symposium on Security and Privacy (Oakland)*, 2017.

[162] Paul Pearce, Ben Jones, Frank Li, Roya Ensafi, Nick Feamster, Nick Weaver, and Vern Paxson. Global measurement of DNS censorship. In *USENIX Security Symposium*, 2017.

[163] Jon Postel. Discard protocol. RFC 863, 1983.

[164] Jon Postel. Echo protocol. RFC 862, 1983.

[165] Jon Postel and Joyce Reynolds. Telnet echo option. RFC 857, 1983.

[166] Prometheus - monitoring system and time series database. Available: https://prometheus.io.

[167] Psiphon. Available: https://psiphon.ca.

[168] Enric Pujol, Oliver Hohlfeld, and Anja Feldmann. Annoyed users: Ads and ad-block usage in the wild. In *ACM Internet Measurement Conference (IMC)*, 2015.

[169] Aravindh Raman, Sagar Joglekar, Emiliano De Cristofaro, Nishanth Sastry, and Gareth Tyson. Challenges in the Decentralised Web: The Mastodon case. In *ACM Internet Measurement Conference (IMC)*, 2019.

[170] Abbas Razaghpanah, Rishab Nithyanand, Narseo Vallina-Rodriguez, Srikanth Sundaresan, Mark Allman, Christian Kreibich, and Phillipa Gill. Apps, trackers, privacy, and regulators: A global study of the mobile tracking ecosystem. In *Internet Society Network and Distributed System Security Symposium (NDSS)*, 2018.

[171] Eric Rescorla and Mozilla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, 2018.

[172] Manoel Horta Ribeiro, Raphael Ottoni, Robert West, Virgílio AF Almeida, and Wagner Meira Jr. Auditing radicalization pathways on youtube. In *ACM Conference on Fairness, Accountability, and Transparency (FAT*)*, 2020.

[173] Florentin Rochet and Olivier Pereira. Waterfilling: Balancing the Tor network with maximum diversity. 2017(2), 2017.

[174] Franziska Roesner, Tadayoshi Kohno, and David Wetherall. Detecting and defending against third-party tracking on the web. In *USENIX Conference on Networked Systems Design and Implementation (NSDI)*, 2012.

[175] University of Oregon Route Views Project. Available: http://www.routeviews.org.

[176] Matthew J. Salganik. *Bit by Bit: Social Research in the Digital Age*. Princeton University Press, 2017.

[177] Quirin Scheitle, Oliver Hohlfeld, Julien Gamba, Jonas Jelten, Torsten Zimmermann, Stephen D Strowes, and Narseo Vallina-Rodriguez. A long way to the top: significance, structure, and stability of internet top lists. In *ACM Internet Measurement Conference (IMC)*, 2018.

[178] Sebastian Schelter, Jérôme Kunegis, et al. On the ubiquity of web tracking: Insights from a billion-page web crawl. *Journal of Web Science*, 4(4), 2018.

[179] Max Schuchard, John Geddes, Christopher Thompson, and Nicholas Hopper. Routing around decoys. In *ACM Conference on Computer and Communications Security (CCS)*, 2012.

[180] Will Scott, Thomas Anderson, Tadayoshi Kohno, and Arvind Krishnamurthy. Satellite: Joint analysis of CDNs and network-level interference. In *USENIX Annual Technical Conference (ATC)*, pages 195–208, 2016.

[181] Robin Seggelmann, Michael Tuexen, and M. Williams. Transport layer security (TLS) and datagram transport layer security (DTLS) heartbeat extension. RFC 6520, 2012.

[182] Ryan Sleevi. Sustaining digital certificate security, October 2015. Available: https://security.googleblog.com/2015/10/sustaining-digital-certificate-security.html.

[183] Drew Springall, Zakir Durumeric, and J Alex Halderman. FTP: The forgotten cloud. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2016.

[184] StartCom will now issue all SSL certificates under 'Certificate Transparency' system,

March 2016. Available: https://www.tomshardware.com/news/startcom-adopts-certificate-transparency-log,31470.html.

[185] Narisu Tao, Xu Chen, and Xiaoming Fu. AS path inference: From complex network perspective. In *IFIP Networking Conference (IFIP Networking)*, 2015.

[186] Willy Tarreau. The PROXY protocol Versions 1 & 2. Available: https://www.haproxy.org/download/1.8/doc/proxy-protocol.txt.

[187] The EasyList authors. EasyList. Available: https://easylist.to/.

[188] Stuart A. Thompson. These ads think they know you. New York Times, April 2019. Available: https://www.nytimes.com/interactive/2019/04/30/opinion/privacy-targeted-advertising.html.

[189] Tor Project. OONI: Open observatory of network interference. Available: https://ooni.torproject.org/.

[190] Vincent Toubiana, Arvind Narayanan, Dan Boneh, Helen Nissenbaum, and Solon Barocas. Adnostic: Privacy preserving targeted advertising. In *Internet Society Network and Distributed System Security Symposium (NDSS)*, 2010.

[191] Tracking the FREAK attack. Available: http://freakattack.com/.

[192] Martino Trevisan, Stefano Traverso, Eleonora Bassi, and Marco Mellia. 4 years of EU cookie law: Results and lessons learned. *Proceedings on Privacy Enhancing Technologies (PoPET)*, 2019(2), 2019.

[193] United States Department of State. Ghana 2016 human rights report, 2016. Available: http://www.state.gov/j/drl/rls/hrrpt/humanrightsreport/index.htm?year=2016&dlid=265260.

[194] Blase Ur, Pedro Giovanni Leon, Lorrie Faith Cranor, Richard Shay, and Yang Wang. Smart, useful, scary, creepy: perceptions of online behavioral advertising. In *USENIX Symposium on Usable Security and Privacy (SOUPS)*, 2012.

[195] Tobias Urban, Dennis Tatang, Martin Degeling, Thorsten Holz, and Norbert Pohlmann. The unwanted sharing economy: An analysis of cookie syncing and user transparency under GDPR, November 2018. Technical Report, arXiv preprint arXiv:1811.08660.

[196] Narseo Vallina-Rodriguez, Johanna Amann, Christian Kreibich, Nicholas Weaver, and Vern Paxson. A tangled mass: The Android root certificate stores. In *ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, 2014.

[197] N Van Eijk, Natalia Helberger, Lars Kool, Arjanna van der Plas, and Bart van der Sloot. Online tracking: Questioning the power of informed consent. *info*, 14, 2012.

[198] Benjamin VanderSloot, Johanna Amann, Matthew Bernhard, Zakir Durumeric, Michael Bailey, and J Alex Halderman. Towards a complete view of the certificate ecosystem. In *ACM Internet Measurement Conference (IMC)*, 2016.

[199] Benjamin VanderSloot, Sergey Frolov, Jack Wampler, Sze Chuen Tan, Irv Simpson, Michalis Kallitsis, J. Alex Halderman, Nikita Borisov, and Eric Wustrow. Running Refraction Networking for real. *Proceedings on Privacy Enhancing Technologies*, 2020(4), 2020.

[200] Benjamin VanderSloot, Allison McDonald, Will Scott, J Alex Halderman, and Roya Ensafi. Quack: Scalable remote measurement of application-layer censorship. In *USENIX Security Symposium*, 2018.

[201] Benjamin VanderSloot, Steven Sprecher, and J. Alex Halderman. Beyond acceptable advertisement: Better understanding blocking extensions, 2020. Under Submission. Available: https://benvds.com/papers/acceptable.pdf.

[202] Antoine Vastel, Pierre Laperdrix, Walter Rudametkin, and Romain Rouvoy. FP-scanner: the privacy implications of browser fingerprint inconsistencies. In *USENIX Security Symposium*, 2018.

[203] Antoine Vastel, Peter Snyder, and Benjamin Livshits. Who filters the filters: Understanding the growth, usefulness and efficiency of crowdsourced ad blocking, 2018. Technical Report, arXiv preprint arXiv:1810.09160.

[204] Giridhari Venkatadri, Elena Lucherini, Piotr Sapiezynski, and Alan Mislove. Investigating sources of PII used in Facebook's targeted advertising. *Proceedings on Privacy Enhancing Technologies (PoPET)*, 2019(1), 2019.

[205] VeriSign: Zone file information. Available: https://www.verisign.com/en_US/channel-resources/domain-registry-products/zone-file/index.xhtml.

[206] Nevena Vratonjic, Julien Freudiger, Vincent Bindschaedler, and Jean-Pierre Hubaux. The inconvenient truth about web certificates. In *Workshop on Economics in Information Security (WEIS)*, 2011.

[207] Robert J Walls, Eric D Kilmer, Nathaniel Lageman, and Patrick D McDaniel. Measuring the impact and perception of acceptable advertisements. In *ACM Internet Measurement Conference (IMC)*, 2015.

[208] Philipp Winter and Stefan Lindskog. How the Great Firewall of China is blocking Tor. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2012.

[209] Emma Woollacott. Apple, Google and Mozilla block Kazakh government surveillance, August 2019. Available: https://www.forbes.com/sites/emmawoollacott/2019/08/21/apple-google-and-mozilla-block-kazakh-government-surveillance/.

[210] Eric Wustrow, Colleen M. Swanson, and J. Alex Halderman. Tapdance: End-to-middle anticensorship without flow blocking. In *USENIX Security Symposium*, 2014.

[211] Eric Wustrow, Scott Wolchok, Ian Goldberg, and J. Alex Halderman. Telex: Anticensorship in the network infrastructure. In *USENIX Security Symposium*, 2011.

[212] Xueyang Xu, Z. Morley Mao, and J. Alex Halderman. Internet censorship in China: Where does the filtering occur? In *Passive and Active Measurement Conference (PAM)*, 2011.

[213] Yaxing Yao, Davide Lo Re, and Yang Wang. Folk models of online behavioral advertising. In *ACM Conference on Computer Supported Cooperative Work and Social Computing (CSCW)*, 2017.

[214] Ting-Fang Yen, Yinglian Xie, Fang Yu, Roger Peng Yu, and Martin Abadi. Host fingerprinting and tracking on the web: Privacy and security implications. In *Internet Society Network and Distributed System Security Symposium (NDSS)*, 2012.

[215] Zhonghao Yu, Sam Macbeth, Konark Modi, and Josep M Pujol. Tracking the trackers. In *International Conference on World Wide Web (WWW)*, 2016.

[216] Savvas Zannettou, Tristan Caulfield, Jeremy Blackburn, Emiliano De Cristofaro, Michael Sirivianos, Gianluca Stringhini, and Guillermo Suarez-Tangil. On the origins of memes by means of fringe web communities. In *ACM Internet Measurement Conference (IMC)*, 2018.

[217] Savvas Zannettou, Tristan Caulfield, William Setzer, Michael Sirivianos, Gianluca Stringhini, and Jeremy Blackburn. Who let the trolls out? towards understanding state-sponsored trolls. In *ACM Conference on Web Science (WebSci)*, 2019.

[218] Liang Zhang, David Choffnes, Dave Levin, Tudor Dumitras, Alan Mislove, Aaron Schulman, and Christo Wilson. Analysis of SSL certificate reissues and revocations in the wake of Heartbleed. In *ACM Internet Measurement Conference (IMC)*, 2014.

[219] Shitong Zhu, Umar Iqbal, Zhongjie Wang, Zhiyun Qian, Zubair Shafiq, and Weiteng Chen. ShadowBlock: A lightweight and stealthy adblocking browser. In *International Conference on World Wide Web (WWW)*, 2019.

[220] Tao Zhu, David Phipps, Adam Pridgen, Jedidiah R Crandall, and Dan S Wallach. The velocity of censorship: High-fidelity detection of microblog post deletions. In *USENIX Security Symposium*, 2013.

[221] Sebastian Zimmeck, Jie S Li, Hyungtae Kim, Steven M Bellovin, and Tony Jebara. A privacy analysis of cross-device tracking. In *USENIX Security Symposium*, 2017.

[222] Jonathan Zittrain and Benjamin Edelman. Internet filtering in China. *IEEE Internet Computing*, 7(2), 2003.