Methods and Systems for Understanding Large-Scale Internet Threats

Paul Pearce

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2018-98
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2018/EECS-2018-98.html

August 5, 2018

Copyright © 2018, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Methods and Systems for Understanding Large-Scale Internet Threats

by

Paul James Pearce

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Vern Paxson, Chair
Professor Deirdre K. Mulligan
Professor Stefan Savage
Professor David Wagner

Summer 2018

Methods and Systems for Understanding Large-Scale Internet Threats

Copyright 2018
by
Paul James Pearce

Abstract

Methods and Systems for Understanding Large-Scale Internet Threats

by

Paul James Pearce

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Vern Paxson, Chair

Large-scale Internet attacks are pervasive. A broad spectrum of actors from organized gangs of criminals to nation-states exploit the modern, layered Internet to launch politically and economically motivated attacks. The impact of these attacks is vast, ranging from billions of users experiencing Internet censorship, to tens of millions of dollars lost annually to cybercrime. Developing effective and comprehensive defenses to these large scale threats requires systematic empirical measurement.

In this dissertation we develop empirical measurement methods and systems for understanding politically and economically motivated Internet threats. Specifically, we examine the problems of Internet censorship and advertising abuse in-depth and at-scale. To understand censorship, we develop Augur and Iris, methods and accompanying systems that allow us to perform global, longitudinal measurement of Internet censorship at the TCP/IP and DNS layers of the network stack—without the use of volunteers. This work addresses a range of both technical and extra-technical challenges, at a scale and fidelity not previously achieved. In combating advertising abuse, we investigate and chronicle multiple facets of the ecosystem— from clickbots to large-scale botnets to advertising injection—using a variety of empirical methods. Our work ultimately identifies fundamental structural weak-points leverageable for defense, resulting in dismantling botnets, cleaning up ad networks, and protecting users.

For Judy

Contents

Acknowledgments

I must begin by thanking my amazing PhD advisor Vern Paxson. His support, understanding, encouragement, and expertise on topics ranging from research to life were invaluable. Without his guidance I would not be where I am today. Thank you.

I must also thank the faculty of the Center for Evidence based Security Research (CESR), namely Stefan Savage and Geoff Voelker. Their council and feedback were instrumental in my career. They both continually went above and beyond in facilitating my PhD, at times engaging more as advisors than mentors. Thank you both.

Similarity, I thank Nick Feamster, whose guidance throughout the 2nd half of my PhD was critical in achieving my career objectives. I also thank David Wagner, a resource and mentor throughout my time in Berkeley.

PhD's are a marathon, and it helps to have good friends to share the load. Thank you to 1044++ for the life part of work-life balance.

I owe a huge debt of gratitude to all of my fellow security students at UC Berkeley and the broader CESR family. This includes but is not limited to: Devdatta Akhawe, Jethro Beekman, Joe DeBlasio, Adrienne Porter Felt, David Fifield, Chris Grier, Grant Ho, Danny Huang, Mobin Javed, Noah Johnson, Frank Li, Bill Marczak, Michael McCoyd, Brad Miller, Ariana Mirian, Austin Murdock, and Kurt Thomas. Thank you all for the support and comradery.

The unsung heros of any PhD are the staff that make everything work. Chief among them are Angie Abbatecola and Jon Kuroda. Thank you both for being awesome and continually going above and beyond in keeping the department going.

Going back further, this degree was not possible without the opportunity of the California Community College system, numerous professors at Chaffey Community College, and the support of several individuals whom facilitated my transfer to UC Berkeley as an undergraduate. Among those who deserve recognition are: Tim Arner (Chaffey Community College Professor, retired), Charles Hollenbeck (Chaffey Community College Professor, retired), Rebecca Miller (former UC Berkeley Director of Student Affairs), Karen Pender (Chaffey Community College Professor, retired), Ana Rafferty (UC Berkeley Transfer Specialist), and Sheila Humphreys (UC Berkeley EECS Emerita Director of Diversity). Thank you all helping me get to this point in my life and career.

Throughout my entire PhD I had the privilege to work with an amazing set of collaborators from both academia and industry. I thank them all for the opportunity to collaborate, and acknowledge that this work was not possible without their assistance.

I also thank my dissertation committee, Vern Paxson, Deirdre Mulligan, Stefan Savage, and David Wagner, for enduring my qualifying exam slides and providing invaluable feedback on my work's direction and impact.

Additionally, I am grateful for the research assistance of the following throughout my graduate career: David Anselmi, Manos Antonakakis, Richard Boscovich, Eric Brewer, Randy Bush, Chia Yuan Cho, Jed Crandall, Hitesh Dharamdasani, Zakir Durumeric, David Fifield, Sarthak Grover, Yunhong Gu, Brad Karp, Niels Provos, Moheeb Abu Rajab, the Google Safe Browsing Team, the

Microsoft Digital Crimes Unit (DCU), the Microsoft Malware Protection Center (MMPC), and the Bing Ads Online Forensics Team.

My work was supported in part by National Science Foundation Awards CNS-0433702, CNS-0831535, CNS-0905631, CNS-1213157, CNS-1237076 CNS-1237264, CNS-1237265, CNS-140-6041, CNS-1518878, CNS-1518918, CNS-1540066, and CNS-1602399; by the Office of Naval Research MURI grants N00014-09-1-1081 and N00014-12-1-0165; by the U.S. Army Research Office MURI grant W911NF-09-1-0553; the Team for Research in Ubiquitous Secure Technology (TRUST); and by gifts from Google, Microsoft, and the UCSD Center for Networked Systems (CNS). Any opinions, findings, and conclusions or recommendations expressed in this dissertation are those of the author and do not necessarily reflect the views of the sponsors.

And to Judy, my wife: You are the best person I know. Your immeasurable patience, love, and support is what allows me to do what I do. Thank you for the journey thus far, and what is come. Also I'm sorry I cold-called you (prior to our formal introduction) when we took EE20N. But it seems to have worked out ok in the long run.

Chapter 1

Introduction

Internet security is defined by conflict. The value and power mediated by the global, interconnected systems of today's Internet in turn attract adversaries who seek to exploit these same systems for economic, political or social gain. However, the underlying complexity of the Internet infrastructure, the layering of its services, and the indirect nature of its business relationships can make it difficult to identify even the *existence* of adversaries manipulating systems for their benefit. Further, identifying that an attack is taking place is only the first step in an ongoing challenge, as adversaries have the luxury to define where and when their actions take place and responders are forced to discover the landscape of the battle after it commences.

Studying these problems is challenging. While anecdotes and serendipitous findings are common, understanding the full nature of a particular action or attack requires systematic measurement—frequently measurement of an actor who seeks to hide or camouflage their actions. Designing *effective* and *comprehensive* defenses requires sound understanding of not only specific problems, but also the fundamental limitations and costs associated with those problems. In the context of global-scale attacks such as cybercrime and censorship, acquiring this understanding is frequently challenging and requires new types of research systems and measurement methods. Remediation without systematic understanding of opponents' costs, capabilities, and objectives risks developing reactionary, incremental defenses lacking feasibility or robustness.

This dissertation brings empirical grounding and understanding to the study of global, hidden Internet security threats. Our work is focused on both politically and economically motivated attacks, spanning censorship (Part I) and cybercrime (Part II).

Part I: Global Censorship Measurement

Anecdotes and reports indicate that Internet censorship is widespread, affecting many countries around the world [55, 112]. Despite this prevalence, empirical Internet measurements revealing the scope and evolution of censorship remain comparatively sparse. This limitation stems from fundamental technical and ethical difficulties in obtaining large-scale, consistent, and sound data, from both numerous countries and multiple vantage points within those countries. Understanding

global manipulation practices is necessary to develop technologies and formulate policies that effectively address censorship.

A more complete understanding of Internet censorship around the world requires *diverse* measurements from a wide range of geographic regions and ISPs, not only across countries but also within regions of a single country. Diversity is important even within countries, because political dynamics can vary internally, and because different ISPs may implement filtering policies differently.

Unfortunately, most mechanisms for measuring Internet censorship globally rely on volunteers who run measurement software deployed on their own Internet-connected devices (e.g., laptops, phones, or tablets) [124, 134]. Because these tools rely on people to install software and perform measurements, it is unlikely that they can ever achieve the scale required to gather continuous and diverse measurements about Internet censorship.

Performing measurements of the scale and frequency necessary to understand the scope and evolution of Internet censorship calls for fundamentally new techniques that do not require human involvement or intervention. Part I of this work develops systems and methods that can perform widespread, ethical, longitudinal measurements of multiple types of global Internet censorship. To achieve the necessary reach and diversity, we have developed systems and methods that do not require the participation of individual users in the countries of interest. This work addresses a range of both technical and extra-technical challenges, at a scale and fidelity not previously achieved.

To enable global continuous measurement of TCP/IP Internet censorship, we develop Augur (Chapter 3) [116, 117], a measurement regimen and accompanying system that soundly leverage potentially highly noisy TCP/IP side channels to measure reachability between two Internet locations without access to measurement vantage points at the locations, or even at points along the path between them (Section 3.2). Augur includes the use of sequential hypothesis testing (Section 3.3) to develop statistical confidence in the face of network and side channel noise, and techniques for responsibly addressing censorship measurement ethics. The validation of Augur included a global censorship measurement study which examined the blocking practices of 179 countries and territories (Section 3.5).

Adversaries employ a variety of technical mechanisms to achieve Internet censorship. Augur enables us to understand global TCP/IP censorship, but to further build a comprehensive picture of censorship, we need techniques to examine additional commonly deployed blocking technologies. Towards this goal we then develop Iris (Chapter 4) [119, 120], a scalable and accurate method to measure global manipulation of DNS resolutions. Iris enables ongoing censorship measurement from 151 countries and territories (Section 4.2). The deployment of Iris reveals new patterns in global censorship, including the heterogeneity of censorship within countries (Section 4.4).

A critical problem in the field of censorship measurement is ethics (Section 2.3). The development of methodologies that are both comprehensive and ethically responsible is challenging, as these goals frequently conflict. By its nature, measuring censorship involves interacting with content that an authority has deemed objectionable; exploring the scope and scale of that content potentially creates risk for those involved in the measurement. Both Augur (Section 3.2) and Iris (Section 4.2) include approaches for reasoning about risk, as well as for reducing potential harm via the use of network infrastructure for indirect censorship measurement.

Part II: Understanding Advertising Abuse

Cybercrime has evolved into a complex global ecosystem of criminal actors performing attacks ranging from ransomware to denial-of-service to click fraud. The motivation for these attacks is economic—criminals carry out these attacks in order to monetize users at a global scale [95, 115, 135]. This economic force means criminals focus their efforts on maximizing profit rather than relying on specific technical mechanisms [136]. Past work has shown that solutions which focus on the financial and relational aspects of cybercrime have the potential to be more effective than incremental technological defenses [95].

One such form of cybercrime, *advertising abuse*, is a prevalent and lucrative attack that exploits the wealth of the online advertising ecosystem while masking criminal identity and activities through the ecosystem's byzantine structure [115, 135]. The scale and structure of these attacks tilt the conflict in favor of the criminals, impacting users on a global scale [115]. Part II of this work systematically examines advertising abuse both from an external perspective and internally through collaborations with industry partners, across multiple monetization strategies and attacks. Using a range of methods and data sources—from command-and-control infiltration, to direct measurement, to industry data—we uncover the scale, structure, and nature of advertising fraud attacks which, by their nature, are designed to be difficult to distinguish. With this understanding, our work identifies fundamental structural weak-points leverageable for defense, resulting in dismantling botnets, cleaning up ad networks, and protecting users.

We begin exploring advertising abuse with an execution-driven study of click fraud malware and the supporting ad ecosystem (Chapter 7 [105]. By iteratively executing malware in isolation with controlled network access, we were able to build tools for automated command-and-control interaction ("milkers"). These tools allowed us to explore the ad abuse ecosystem at a scale not possible with traditional malware execution. This exploration uncovered the breath and scale of the ad fraud ecosystem, as well as the fundamentals of the business model. Despite this new understanding of the ecosystem, our view was still limited to an external ad placement perspective.

To obtain a view of the ad fraud ecosystem at both a larger scale and from an internal perspective, we next perform an in-depth exploration of ZeroAccess (ZA) in Chapter 8. ZA was a vast and complex peer-to-peer (P2P) botnet, serving as a delivery platform for advertising abuse malware for more than four years. At its peak, ZA infected more than 1.9 million systems, resulting in millions of dollars in advertising fraud per month.

Continuing to explore the advertising fraud ecosystem, we then develop an in-depth technical analysis (Chapter 9) on the structure and function of the ZA malware [118]. This work led to collaboration with Microsoft's Digital Crime Unit and law enforcement, with the technical analysis serving as Exhibit 1 in legal action against the criminal actors [104].

Chapters 7, 8, and 9 provide an external view of the structure and function of large-scale advertising abuse. In order to have a qualitatively different, insider view of ad fraud, in Chapter 10 we combine an array of data sources, including P2P measurements, command-and-control telemetry from botnet infiltration, along with click information from a top ad-network industry partner. This work illuminated the rich, intertwined nature of malware-driven click fraud and the advertising ecosystem it exploited, as well as how to develop effective remediations [115]. Using this

multifaceted approach, we identify fraudulent business relationships within the advertising network stemming from complex multi-hop ad reseller chains, which were used as a focal point for remediation. We also quantified the financial impact of ZA's criminal activity.

A key result from from Chapters 9 and 10 is the complexity of multi-hop ad reseller chains, and how those chains can be used to mask and "launder" fraud. Chapter 11 explores this issue specifically, focusing on a different facet of the advertising abuse ecosystem—"ad injection"—which affected *tens of millions* of users [135]. Similar to malware-driven click fraud, ad injection generates revenue through a complex and intertwined ecosystem of intermediaries with opaque business relationships used to launder ad views and clicks. But in the case of ad injection, software that is likely unwanted or has misrepresented its purpose injects ads into the browsing experiences of actual users. Via a collaboration with Google, Chapter 11 explores the structure and composition of traffic intermediaries and advertisers that served as the revenue source feeding the injection ecosystem. Our work enabled remediation for millions of users by identifying structural "choke points" of three ad networks and 25 affiliate programs that were responsible for the majority of ad injections.

Part I

Global Censorship Measurement

Chapter 2

Censorship Introduction, Related Work and Ethics

2.1 Introduction

News reports, anecdotes, and policy briefings collectively suggest that Internet censorship—the blocking and manipulation of content deemed to be objectionable by controlling entities—is pervasive [55]. Despite this prevalence, the nature of Internet censorship and the continuous evolution of how and where censorship is applied, measurements of censorship remain comparatively sparse. The scale and diversity of Internet censorship practices makes it difficult to precisely monitor where, when, and how censorship occurs, as well as what is censored. The potential risks in performing the measurements make this problem even more challenging. As a result, many accounts of censorship begin—and end—with anecdotes or short-term studies from only a handful of vantage points.

Understanding the scope, scale, and evolution of Internet censorship requires global measurements, performed at regular intervals. Unfortunately, the state of the art relies on techniques that, by and large, require users to directly participate in gathering these measurements [71, 124, 134], drastically limiting their coverage and inhibiting regular data collection.

These approaches remain difficult to deploy in practice: for example, some countries might not have globally available VPN exits within them that some techniques rely upon [71], or may have censors that block the network access required for the measurements [134]. Because such approaches and tools rely on people to install software and perform measurements, it is unlikely that they can ever achieve the scale required to gather continuous and diverse measurements about Internet censorship.

Another approach is to opportunistically leverage a network presence in a given country using browser-based active measurement of potential censorship [128]. This method can have difficulties in obtaining fully global views, though, because it is driven by end-user browsing choices. Due to its potential for implicating end users in attempting to access prohibited Internet sites, it can only be used broadly to measure reachability to sites that would pose minimal additional risk to users,

which limits its utility for measuring reachability to a broad range of sites.

Part I seeks to instead develop methods and tools to continuously monitor information about Internet reachability, remotely, without the need for volunteers or user participation (willing or unwilling). Such methods have the capability to capture the onset or termination of censorship across regions and ISPs, as well as identify differences in censorship behaviors across both time and space.

To achieve these goals we first introduce *Augur* (Chapter 3) [116, 117], a method and accompanying system that utilizes IP spoofing and TCP/IP side channels to measure reachability between two Internet locations without directly controlling a measurement vantage point at either location. Using these side channels, coupled with techniques to ensure safety by not implicating individual users, we develop scalable, statistically robust methods to infer network-layer filtering, and implement a corresponding system capable of performing continuous monitoring of global censorship. We validate our measurements of Internet-wide disruption in 179 countries (and territories) over 17 days against sites known to be frequently blocked; we also identify the countries where connectivity disruption is most prevalent.

Next, we develop Iris (Chapter 4) [119, 120], a scalable and accurate method to measure global manipulation of DNS resolutions using open infrastructure DNS resolvers. Iris facilitates large-scale measurements that can expand our understanding of censorship beyond Augur's IP-level view. Iris reveals widespread DNS manipulation of many domain names; our findings both confirm anecdotal reports and results from previous work, and reveal new patterns in DNS manipulation.

2.2 Related Work

Previous work spans several related areas. We begin discussing previous research which has performed pointwise studies of censorship in specific countries, as well as tools that researchers have developed to facilitate global censorship measurement studies. Next, we discuss previous studies that have highlighted the variability and volatility of censorship measurements over time and across regions, which motivates our work. Finally we conclude with discussion of closely related work on connectivity measurements using side channels, DNS manipulation, and open resolvers.

Country-specific censorship studies. In recent years many researchers have investigated the whats, hows, and whys of censorship in particular countries. These studies often span a short period of time and reflect a single vantage point within a target country, such as by renting virtual private servers. For example, studies have specifically focused on censorship practices in China [7, 31, 151, 154], Iran [11], Pakistan [82, 108], Syria [23], and Egypt [12]. Studies have also explored the employment of various censorship methods, e.g., country-wide Internet outages [35], injection of fake DNS replies [9, 97], blocking of TCP/IP connections [148], and application-level blocking [36, 76, 114], and traffic throttling [4]. A number of studies suggest that countries sometimes change their blocking policies and methods in times surrounding political events. For example, Freedom House reports 15 instances of Internet shutdowns—where the government cut off access to Internet entirely—in 2016 alone [55]. Most of these were apparently intended to prevent

citizens from reaching social media to spread prohibited information. In general, studies involving direct measurements can shed more light on specific mechanisms that a censor might employ. By contrast, the techniques we develop rely on indirect side channels and open DNS resolvers, which limits the types of measurements that we can perform. On the other hand, our approach permits a much larger scale than any of these previous studies, as well as the ability to conduct measurements continuously. Although these studies provide valuable insights, their scale often involves a single vantage point for a limited amount of time (typically no more than a few weeks). Our aim is to shed light on a much broader array of Internet vantage points, continuously over time.

Global censorship measurement tools. Several research efforts developed platforms to measure censorship by running experiments from diverse vantage points. For instance, CensMon [131] used PlanetLab nodes in different countries, and UBICA [2] aimed to increase vantage points by running censorship measurement software on home gateway devices and user desktops. In practice, as far as we know, neither of these frameworks are still deployed and collecting data. The OpenNet Initiative [112] has used its public profile to recruit volunteers around the world who have performed one-off measurements from home networks each year for the past ten years. OONI [134] and ICLab [71], two ongoing data collection projects, use volunteers to run both custom software and custom embedded devices (such as Raspberry Pis [53]).

Although each of these frameworks can perform an extensive set of tests, they rely on volunteers who run measurement software on their Internet-connected devices. These human involvements make it more challenging—if not impossible—to gather continuous and diverse measurements.

Studies that highlight the temporal and spatial variability of connectivity disruptions. If patterns of censorship and connectivity disruptions hold relatively static, then existing one-off measurement studies would suffice to over time build up a global picture of conditions. Previous work, however, has demonstrated that censorship practices vary across time; across different applications; and across regions and Internet service providers, even within a single country. For example, previous research found that governments target a variety of services such as video portals (e.g., YouTube) [140], news sites (e.g.,`bbc.com`) [13], and blogs (e.g., `livejournal.com`) [5]. Censors also target circumvention and anonymity tools; most famously, the Great Firewall of China has engaged in a decade-long cat-and-mouse game with Tor [49, 147, 148]. Ensafi [46] showed that China's Great Firewall (GFW) actively probes—and blocks upon confirmation—servers suspected to abet circumvention. Many studies show that different countries employ different censorship mechanisms beyond IP address-blocking to censor similar content or applications, such as Tor [139]. Occasionally, countries also deploy new censorship technology shortly before significant political events. For example Aryan [11] studied censorship in Iran before and after the June 2013 presidential election. Although these studies provide important data points, each reflects a snapshot at a single point in time and thus cannot capture ongoing trends and variations in censorship practices.

Measuring connectivity disruptions with side channels. Previous work has employed side channels to infer network properties such as topology, traffic usage, or firewall rules between two remote hosts [24, 48, 49, 123, 153]. Some of these techniques rely on the fact that the IP identifier (IP ID) field can reveal network interfaces that belong to the same Internet router, the number of packets that a device generates [24], or the blocking direction of mail server ports for anti-spam purposes [123].

The SYN backlog also provides another signal that helps with the discovery of machines behind firewalls [48, 153]. Ensafi et al. [47] observed that combining information from the TCP SYN backlog (which initiates retransmissions of SYN ACK packets) with IP ID changes can reveal packet loss between two remote hosts, including the direction along the path where packet drops occurred; the authors demonstrated the utility of their technique by measuring the reachability of Tor relays from China [49]. Our work builds on this technique by developing robust statistical detection methods to disambiguate connectivity disruptions from other effects that induce signals in these side channels.

Measuring DNS manipulation. The DNS protocol's lack of authentication and integrity checking makes it a prime target for attacks. Jones et al. presented techniques for detecting unauthorized DNS root servers, though found little such manipulation in practice [75]. Jiang et al. identified a vulnerability in DNS cache update policies that allows malicious domains to stay in the cache even if removed from the zone file [73].

Several projects have explored DNS manipulation using a limited number of vantage points. Weaver et al. explored DNS manipulation with respect to DNS redirection for advertisement purposes [144]. The authors also observed incidents in which DNS resolvers redirected end hosts to malware download pages. There are many country-specific studies that show how different countries use a variety of DNS manipulation techniques to exercise Internet censorship. For example, in Iran the government expects ISPs to configure their DNS resolvers to redirect contentious domains to a censorship page [11]. In Pakistan, ISPs return NXDOMAIN responses [108]. In China, the Great Firewall injects forged DNS packets with seemingly arbitrary IP addresses [9]. These studies however all drew upon a small or geographically limited set of vantage points, and for short periods of time.

Using open resolvers. A number of studies have explored DNS manipulation at a larger scale by probing the IPv4 address space to find open resolvers. In 2008, Dagon et al. found corrupt DNS resolvers by running measurements using 200,000 open resolvers [34]; they do not analyze the results for potential censorship. A similar scan by anonymous authors [8] in 2012 showed evidence of Chinese DNS censorship affecting non-Chinese systems.

Follow-on work in 2015 by Kührer et al. tackled a much larger scope: billions of lookups for 155 domain names by millions of open resolvers [93]. The study examined a broad range of potentially tampered results, which in addition to censorship included malware, phishing, domain parking, ad injection, captive portals, search redirection, and email delivery. They detected DNS manipulation by comparing DNS responses from open resolvers with ground truth resolutions

gathered by querying control resolvers. They then identified legitimate unmanipulated answers using a number of heuristic filtering stages, such as treating a differing response as legitimate if its returned IP address lies within the same AS the ground truth IP address.

We initially sought to use their method for conducting global measurements specifically for detecting censorship. However, censorship detection was not a focus of their work, and the paper does not explicitly describe the details of its detection process. In particular, other than examining HTTP pages for "blocked by the order of ..." phrasing, the paper does not present a decision process for determining whether a given instance of apparent manipulation reflects censorship or some other phenomenon. In addition, their measurements leverage open resolvers *en masse*, which raises ethical concerns for end users who may be wrongly implicated for attempting to access banned content. In contrast, we frame a detailed, reproducible method for globally measuring DNS-based manipulation in an ethically responsible manner.

In 2016, Scott et al. introduced Satellite [130], a system which leverages open resolvers to identify CDN deployments and network interference using collected resolutions. Given a bipartite graph linking domains queried with IP address answers collected from the open resolvers, Satellite identifies strongly connected components, which represent domains hosted by the same servers. Using metrics for domain similarity based on the overlap in IP addresses observed for two domains, Satellite distinguishes CDNs from network interference as components with highly similar domains (additionally, other heuristics help refine this classification).

2.3 Ethics

The design of our censorship measurement methods incorporates many considerations regarding ethics. Our primary ethical concern is the risks associated with the measurements that Augur and Iris conduct, as leveraging infrastructure we do not control potentially implicates otherwise innocent users. A second concern is whether the measurements we generate introduce undue query load on various pieces of Internet infrastructure. With these concerns in mind, we consider the ethics of performing our measurements using the ethical guidelines of the Belmont Report [14] and Menlo Report [41] to frame our discussion.

Additional information on the specific application of these guidelines to teach technique can be found in Section 3.2 for Augur and Section 4.2 for Iris.

One important ethical principle is *respect for persons*; essentially, this principle states that an experiment should respect the rights of humans as autonomous decision-makers. Sometimes this principle is misconstrued as a requirement for informed consent for all experiments. In many cases, however, informed consent is neither practical nor necessary; accordingly, Salganik [127] characterizes this principle instead as "some consent for most things". In the case of Augur and Iris, obtaining the consent of parties is impractical.

In lieu of attempting to obtain informed consent, we turn to the principle of *beneficence*, which weighs the benefits of conducting an experiment against the risks associated with the experiment. Note that the goal of beneficence is not to *eliminate* risk, but merely to *reduce* it, and then ensure

the benefits out-weigh the risks. The design of Augur (Section 3.2) and Iris (Section 4.2) rely heavily on this principle.

The principle of *justice* states that the beneficiaries of an experiment should be the same population that bears the risk of that experiment. On this front, we envision that the beneficiaries of the kinds of measurements that we collect using Augur and Iris will be wide-ranging: designers of circumvention tools, as well as policymakers, researchers, and activists who are improving communications and connectivity for citizens in oppressive regimes all need better data about the extent and scope of Internet censorship.

A final guideline concerns *respect for law and public interest*, which essentially extends the principle of beneficence to all relevant stakeholders, not only the experiment participants. This principle is useful for reasoning about the externalities that our measurements create by increasing load on Internet infrastructure for various DNS domains and IP addresses. To abide by this principle, we rate-limit our measurements to ensure that the owners of these domains and IPs do not face large expenses as a result of the queries that we issue.

Chapter 3

Augur: Internet-Wide Detection of Connectivity Disruptions

3.1 Introduction

Advances in TCP/IP side-channel measurement techniques offer a new paradigm for obtaining global-scale visibility into Internet connectivity. Ensafi et al. developed *Hybrid-Idle Scan*, a method whereby a third vantage point can determine the state of network-layer reachability between two other endpoints [47]. This allows an off-path measurement system to infer whether two remote systems can communicate with one another, regardless of where these two remote systems are located. To perform these measurements, the off-path system must be able to spoof packets (i.e., it must reside in a network that does not perform egress filtering), and one of the two endpoints must use a single shared counter for generating the IP identifier value for packets that it generates. This technique provides the possibility of measuring network-layer reachability around the world by locating endpoints within each country that use a shared IP ID counter. By measuring the progression of this counter over time, as well as whether our attempts to perturb it from other locations on the Internet, we can determine the reachability status between pairs of Internet endpoints. This technique makes it possible to conduct measurements continuously, across a large number of vantage points.

Despite the conceptual appeal of this approach, realizing the method poses many challenges. One challenge concerns *ethics*: Using this method can make it appear as though a user in some country is attempting to communicate with a potentially censored destination, which could imperil users. To abide by the ethical guidelines set out by the Menlo [41] and Belmont [14] reports, we exercise great care to ensure that we perform our measurements from Internet infrastructure (e.g., routers, middleboxes), as opposed to user machines. A second challenge concerns *statistical robustness* in the face of unrelated network activity that could interfere with the measurements, as well as other systematic errors concerning the behavior of TCP/IP side channels that sometimes only become apparent at scale. To address these challenges we introduce *Augur*. To perform detection in the face of uncertainty, we model the IP ID increment over a time interval as a random

variable that we can condition on two different priors: with and without responses to our attempts to perturb the counter from another remote Internet endpoint. Given these two distributions, we can then apply statistical hypothesis testing based on maximum likelihood ratios.

We validate our Augur measurements of Internet-wide disruption in nearly 180 countries (and dependent territories) over 17 days against both block lists from other organizations as well as known IP addresses for Tor bridges. We find that our results are consistent with the expected filtering behavior from these sites. We also identify the top countries that experience connectivity disruption; our results highlight many of the world's most infamous censors.

We begin in Section 3.2 with an overview of our method. We present Augur in Section 3.3, introducing the principles behind using IP ID side channels for third-party measurement of censorship; discussing how to identify remote systems that enable us to conduct our measurements in an ethically responsible manner; and delving into the extensive considerations required for robust inference. In Section 3.4, we present a concrete implementation of Augur. In Section 3.5, we validate Augur's accuracy and provide an accompanying analysis of global censorship practices observed during our measurement run. We offer thoughts related to further developing our approach in Section 3.6 and summarize in Section 3.7.

This chapter is based on work that appeared in the IEEE Symposium on Security and Privacy (S&P) [116] and IEEE Security & Privacy Magazine [117].

3.2 Method Overview

In this section, we provide an overview of the measurement method that we developed to detect filtering. We frame the design goals that we aim to achieve and the core technique underlying our approach. Then in Section 3.3 we provide a detailed explanation of the system's operations.

Design Goals

We first present a high-level overview of the strategy underlying our method, which we base on inducing and observing potential increments in an Internet host's IP ID field. The technique relies on causing one host on the Internet to send traffic to another (potentially blocked) Internet destination; thus, we also consider the ethics of the approach. Finally, we discuss the details of the method, including how we select the specific Internet endpoints used to conduct the measurements.

Ultimately, the measurement system that we design should achieve the following properties:

- *Scalable.* Because filtering can vary across regions or ISPs within a single country, the system must be able to assess the state of filtering from a large number of vantage points. Filtering will also vary across different destinations, so the system must also be able to measure filtering to many potential endpoints.
- *Efficient.* Because filtering practices change over time, establishing regular baseline measurements is important, to expose transient, short-term changes in filtering practices, such as those that might occur around political events.

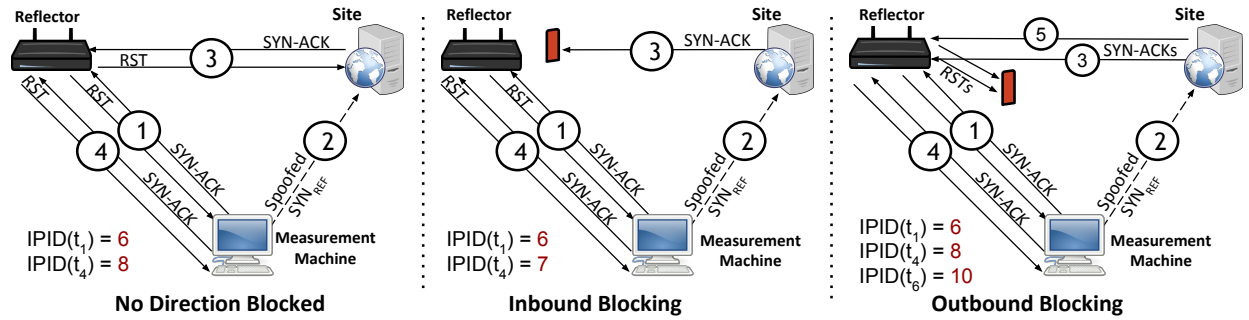


Figure 3.1: Overview of the basic method of probing and perturbing the IP ID side channel to identify filtering. Reflectors are hosts on the Internet with a global IP ID. Sites are potentially filtered hosts that respond to SYN packets on port 80. (In the right hand figure, we omit subsequent measuring of the reflector's IP ID by the measurement machine at time t_6). Spoofed SYN packets have a source field set to the reflector.

- *Sound.* The technique should avoid false positives and ensure that repeated measurements of the same phenomenon produce the same outcome.
- *Ethical.* The system design must satisfy the ethical principles from the Belmont [14] and Menlo [41] Reports: respect for people, beneficence, justice, and respect for law and public interest.

We present a brief overview of the scanning method before explaining how the approach satisfies the design goals above.

Approach

The strategy behind our method is to leverage the fact that when an Internet host generates and sends IP packets, each generated packet contains a 16-bit IP identifier ("IP ID") value that is intended to assist endpoints in re-assembling fragmented IPv4 packets. Although path MTU discovery now largely obviates the need for IP fragmentation, senders still generate packets with IP ID values. There are only 2^{16} unique IP ID values, but the intent is that subsequent packets from the same host should have different IP ID values.

When an Internet host generates a packet, it must determine an IP ID to use for that packet. Although different hosts on the Internet use a variety of mechanisms to determine the IP ID for each packet (e.g., random, counter-based increment per-connection or per-interface), many hosts use a single global counter to increment the IP ID value for all packets that originate from that host, regardless of whether the packets it generates bear a relationship to one another. In these cases where the host uses a single IP ID counter, the value of the counter at any time reflects how many packets the host has generated. Thus, the ability to observe this counter over time gives an indication of whether a host is generating IP packets, and how many.

The basic method involves two mechanisms:

- *Probing:* A mechanism to observe the IP ID value of a host at any time.
- *Perturbation:* A mechanism to send traffic to that same host from different Internet destinations, which has the property of inducing the initial host to respond, thus incrementing its IP ID counter.

We now describe the basic design for probing and perturbation, in the absence of various complicating factors such as cross-traffic or packet loss. Figure 3.1 illustrates the process.

To *probe* the IP ID value of some host over time, a measurement machine sends unsolicited TCP SYN-ACK packets to the host and monitors the responses—TCP RST packets—to track the evolution of the host's IP ID. We monitor the IP ID values at the host on one end of the path. We call this host the *reflector*, to denote that the host reflects RST packets from both our measurement machine and the endpoint that a censor may be trying to filter. This reflector is a machine in a network that may experience IP filtering. We call the other endpoint of this connection the *site*, as for our purposes we will commonly use for it a website operating on port 80.

To *perturb* the IP ID values on either end of the path, a measurement machine sends a TCP SYN packet to one host, the site; the TCP SYN packet carries the (spoofed) source IP address of a second machine, the reflector. We term this *injection*. If no filtering is taking place, the SYN packet from the measurement machine to the site will elicit a SYN-ACK from the site to the reflector, which will in turn elicit a RST from the reflector to the site (since the reflector had not previously sent a TCP SYN packet for this connection). When the reflector sends a RST packet to the site, it uses a new IP ID. If the reflector generates IP ID values for packets based on a single counter, the measurement machine can observe whether the reflector generated a RST packet with subsequent probes, because the IP ID counter will have incremented by two (one for the RST to the site, one for the RST to our measurement machine). Figure 3.1 shows this process in the "no direction blocked" scenario.

Suppose that filtering takes place on the path between the site and the reflector (i.e., one of the other two cases shown in Figure 3.1). We term blocking that manifests on the path from the site to the reflector as *inbound* blocking. In the case of inbound blocking, the site's SYN-ACK packet will not reach the origin, thus preventing the expected IP ID increment at the reflector. In the absence of other traffic, the IP ID counter will increment by one. We show this in the second section of Figure 3.1.

Conversely, we call blocking on the path from the reflector to the site *outbound* blocking; in the case of outbound blocking, SYN-ACK packets from the site reach the reflector, but the RST packets from the reflector to the site never reach the site. At this point, the site should continue to retransmit SYN-ACK packets [138], inducing further increments in the IP ID value at the reflector at various intervals, though whether and how it actually does so depends on the configuration and specifics of the site's operating system. The final section of Figure 3.1 shows the retransmission of SYN-ACK packets and the increment of the global IP ID at two different times. If our measurements reveal a site as inbound-blocked, filtering may actually be bidirectional. We cannot differentiating between the two using this technique because there is no way to remotely induce the reflector to send packets to the site.

Ethics

The measurement method we develop generates spoofed traffic between the reflector and the site which might cause an inexperienced observer of these measurements to (wrongly) conclude that the person who operates or owns the reflector was willfully accessing the site. The risks of this type of activity are unknown, but are likely to vary by country. Although the spoofed nature of the traffic is similar to common large-scale denial-of-service backscatter [107] and results in no data packets being exchanged between reflector and site, we nonetheless use extreme caution when selecting each reflector. In this type of measurement, we must first consider *respect for humans*, by limiting the potential harm to any person as a result of this experiment. One mechanism for demonstrating respect for humans is to obtain informed consent; unfortunately, obtaining informed consent is difficult, due to the scope, scale, and expanse of the infrastructure that we employ.

Salganik explains that the inability to obtain informed consent does not by itself reflect a disregard of respect for humans [127]. Rather, we must take other appropriate measures to ensure that we are abiding by the ethical principles from the Belmont [14] and Menlo [41] reports. To do so, we develop a method that reduces the likelihood that we are directly involving any humans in our experiments in the first place, by focusing our measurements on *infrastructure*. Specifically, our method works to limit the endpoints that we use as reflectors to likely Internet infrastructure (e.g., routers in the access or transit networks, middleboxes), as opposed to hosts that belong to individual citizens (e.g., laptops, desktops, home routers, consumer devices). To do so, we use the CAIDA Ark dataset [22], which contains traceroute measurements to all routed /24 networks. We include a reflector in our experiments only if it appears in an Ark traceroute at least two hops away from the traceroute endpoint. The Ark dataset is not comprehensive, as the traceroute measurements are conducted to a randomly selected IP address in each /24 prefix. Restricting the set of infrastructure devices to those that appear in Ark restricts the IP addresses we might be able to discover with a more comprehensive scan.

Although this approach increases the likelihood that the reflector IP addresses are routers or middleboxes as opposed to endpoints, the method is not fool-proof. For example, devices that are attributable to individuals might still be two hops from the network edge, or a network operator might be held accountable for the *perceived* actions performed by the machines. Our techniques do not eliminate risk. Rather, in accordance with the ethical guideline of *beneficence*, they reduce it to the point where the benefits of collecting these measurements may outweigh the risks of collecting them. In keeping with Salganik's recommendations [127], we aim to conduct measurements that pose a *minimal additional risk*, given both the nature of the spoofed packets and the potential benefits of the research.

The Internet-wide scans we conduct using ZMap [44] to detect possible reflectors introduce concerns related to *respect for law and public interest*. Part of the respect of law and public interest is to reduce the network load we induce on reflectors and sites, to the extent possible, as unnecessary network load could drive costs higher for the operators of reflectors and sites; if excessive, the probing traffic could also impede network performance. To mitigate these possible effects, we follow the approach for ethical scanning behavior as outlined by Durumeric et al. [44]: we signal the benign intent of our scans in the WHOIS entries and DNS records for our scanning

IPs, and provide project details on a website hosted on each scanning machine. We extensively tested our scanning methods prior to their deployment; we also respect opt-out requests.

The measurement probes and perturbations raise similar concerns pertaining to respect for law and public interest. We defer the details of the measurement approach to Section 3.3 but note that reflectors and sites receive an average of one packet per second, with a maximum rate of ten SYN packets in a one-second interval. This load should be reasonable, given that reflectors represent Internet infrastructure that should be able to sustain modest traffic rates directed to them, and sites are major websites that see much higher traffic rates than those we are sending. To ensure that our TCP connection attempts do not use excessive resources on sites or reflectors, we promptly reset any half-open TCP connections that we establish.

The ethical principle of *justice* states that the parties bearing the risk should be the same as those reaping the benefits; the parties who would bear the risk (users in the countries where censorship is taking place) may ultimately reap some benefit from the knowledge about filtering that our tools provide through improved circumvention tools and better information about what is blocked.

3.3 Putting the Method to Practice

In this section, we present our approach for identifying reflectors and sites, and then develop in detail how we perform the measurements described in Section 3.2.

Reflector Requirements

Suitable reflectors must satisfy four requirements:

1. **Infrastructure machine.** To satisfy the ethical guidelines that we outlined in Section 3.2, the reflector should be Internet infrastructure, as opposed to a user machine.

2. **RST packet generation.** Reflectors must generate TCP RST packets when receiving SYN-ACKs for unestablished connections. The RST packets increment the reflector's IP ID counter while ensuring that the site terminates the connection.

3. **Shared, monotonically incrementing IP ID.** If a reflector uses a shared, monotonic strictly increasing per-machine counter to generate IP ID values for packets that it sends, the evolution of the IP ID value—which the measurement machine can observe—will reflect any communication between the reflector and any other Internet endpoints.

4. **Measurable IP ID perturbations.** Because the IP ID field is only 16 bits, the reflector must not generate so much traffic so as to cause the counter value to frequently wrap around between successive measurement machine probes. The natural variations of the IP ID counter must also be small compared to the magnitude of the perturbations that we induce.

Section 3.4 describes how we identify reflectors that meet these requirements.

Site Requirements

Our method also requires that sites exhibit certain network properties, allowing for robust measurements at reflectors across the Internet. Unlike reflectors, site requirements are not absolute. In some circumstances, failure to meet a requirement requires discarding of a result, or limits possible outcomes, but we can still use the site for some measurements.

1. **SYN-ACK retransmission (SAR).** SYN-ACK retries by sites can signal outbound blocking due to a reflector's RST packets not reaching the site. If a site does not retransmit SYN-ACKs, we can still detect inbound blocking, but we cannot distinguish instances of outbound blocking from cases where there is no blocking.

2. **No anycast.** If a site's IP address is anycast, the measurement machine and reflector may be communicating with different sites; in this case, RSTs from the reflector will not reach the site that our measurement machine communicates with, which would result in successive SYN-ACK retransmissions from the site and thus falsely indicate outbound blocking.

3. **No ingress filtering.** If a site's network performs ingress filtering, spoofed SYN packets from the measurement machine may be filtered if they arrive from an unexpected ingress, falsely indicating inbound blocking.

4. **No stateful firewalls or network-specific blocking.** If a site host or its network deploys a distributed stateful firewall, the measurement machine's SYN packet may establish state at a different firewall than the one encountered by a reflector's RSTs, thus causing the firewall to drop the RSTs. This effect would falsely indicate outbound blocking. Additionally, if a site or its firewall drops traffic from some IP address ranges but not others (e.g., from non-local reflectors), the measurement machine may falsely detect blocking.

Section 3.4 describes how we identify sites that satisfy these requirements.

Detecting Disruptions

As discussed in Section 3.2, we detect connectivity disruptions by perturbing the IP ID counter at the reflector and observing how this value evolves with and without our perturbation.

Approach: Statistical detection. We measure the natural evolution of a reflector's counter periodically in the absence of perturbation as a control that we can compare against the evolution of the IP ID under perturbation. We then perturb the IP ID counter by injecting SYN packets and subsequently measure the evolution of this counter. We take care not to involve any site or reflector in multiple simultaneous measurements, since doing so could conflate two distinct results.

Ultimately, we are interested in detecting whether the IP ID evolution for a reflector changes as a result of the perturbations we introduce. We can represent this question as a classical problem in statistical detection, which attempts to detect the presence or absence of a prior (i.e., perturbation

or no perturbation), based on the separation of the distributions under different values of the prior. In designing this detection method, we must determine the random variable whose distribution we wish to measure, as well as the specific detection approach that allows us to distinguish the two values of the prior with confidence. We choose IP ID *acceleration* (i.e., the second derivative of IP ID between successive measurements) as ideally this value has a zero mean, regardless of reflector. With a zero mean, the distribution of the random variable should be stationary and the distribution should be similar across reflectors. Conceptually, this can be thought of as a reflector, at a random time, being as likely to experience traffic "picking up" as not. However, subtle Internet complexities such as TCP slow start bias this measure slightly. We discuss empirical measures of these priors and their impact on our method in Section 3.4.

In contrast, the first derivative (IP ID velocity) is not stationary. Additionally, each reflector would exhibit a different mean velocity value, requiring extensive per-reflector baseline measurements to capture velocity behavior.

Detection framework: Sequential hypothesis testing (SHT). We use sequential hypothesis testing (SHT) [78] for the detection algorithm. SHT is a statistical framework that uses repeated trials and known outcome probabilities (*priors*) to distinguish between multiple hypotheses. The technique takes probabilities for each prior and tolerable false positive and negative rates as input and performs repeated online trials until it can determine the value of the prior with the specified false positive and negative rates. SHT's ability to perform online detection subject to tunable false positive/negative rates, and its tolerance to noise, make it well-suited to our detection task. Additionally, it is possible to compute an expectation for the number of trials required to produce a detection, thus enabling efficient measurement.

We begin with the SHT formulation developed by Jung et al. [78], modifying it to accommodate our application. For this application to hold, the IP ID acceleration must be stationary (discussed more in Section 3.4), and the trials must be independent and identically distributed (i.i.d.). To achieve i.i.d., we randomize our trial order and mapping between sites and reflectors and run experiments over the course of weeks.

For a given site S_i and reflector R_j, we perform a series of N trials, where we inject spoofed SYN packets to S_i and observe IP ID perturbations at R_j. We let $Y_n(S_i,R_j)$ be a random variable for the nth trial, such that:

$$Y_n(S_i,R_j) = \begin{cases} 0 & \text{if no IP ID acceleration occurs} \\ 1 & \text{if IP ID acceleration occurs} \end{cases}$$

during the measurement window following injection. We identify two hypotheses: H_0 is the hypothesis that no inbound blocking is occurring (the second derivative of IP ID values between successive measurements should be observed to be positive, which we define as IP ID acceleration), and H_1 is the hypothesis that blocking is occurring (no IP ID acceleration). Following constructions from previous work, we must identify the prior conditional probabilities of each hypothesis, specifically:

$$\Pr[Y_n = 0|H_0] = \theta_0, \quad \Pr[Y_n = 1|H_0] = 1 - \theta_0$$

$$\Pr[Y_n = 0|H_1] = \theta_1, \quad \Pr[Y_n = 1|H_1] = 1 - \theta_1$$

The prior θ_1 is the probability of no observed IP ID acceleration in the case of inbound blocking. We can experimentally measure this prior as the probability of IP ID acceleration during our reflector control measurements, since the IP ID acceleration likelihood during control measurements is the same as during inbound blocking (as no additional packets reach the reflector in both cases). Intuitively, we can think of this value as 0.5 given the prior discussion of second-order value being thought of as zero mean (i.e., in aggregate traffic, with no induced behavior, acceleration is as likely to occur as deceleration).

The prior $1 - \theta_0$ is the probability of observed IP ID acceleration during injection. It can be measured as the probability of IP ID acceleration during an injection period across all reflector injection measurements. Assuming no blockage and perfect reflectors with no other traffic, this value can be thought of as approaching 1. The prior can be estimated from all reflector measurements under the assumption that blocking is uncommon for a reflector. However, even if the assumption does not hold and blocking is common, the prior estimation is still *conservative* in that it drives the prior closer to the θ_1, making detection more difficult, increasing *false negatives*.

From the construction above, we define a likelihood ratio $\Lambda(Y)$, such that:

$$\Lambda(Y) \equiv \frac{\Pr[Y|H_1]}{\Pr[Y|H_0]} = \Pi_{n=1}^{N} \frac{\Pr[Y_n|H_1]}{\Pr[Y_n|H_0]}$$

where Y is the sequence of trials observed at any point. We derive an upper bound threshold η_1 such that:

$$\frac{\Pr[Y_1, \ldots Y_N|H_1]}{\Pr[Y_1, \ldots Y_N|H_0]} \geq \eta_1$$

and a similar lower bound threshold η_0. Both η_0 and η_1 are bounded by functions of the tolerable probability of false positives and negatives. We elaborate on these bounds and the impact of false positives and negatives later in this section.

Figure 3.2 illustrates our detection algorithm, which performs a series of sequential hypothesis tests; the rest of this section describes this construction in detail. The Inbound Blocking portion of Figure 3.2 shows how SHT uses this construction to make decisions. This is extended to include outbound blocking subsequently.

As each trial is observed, we update the likelihood ratio function $\Lambda(Y)$ based on the prior probabilities. Once updated, we compare the value of $\Lambda(Y)$ against the thresholds η_0 and η_1. If $\Lambda(Y) \leq \eta_0$, we accept H_1 and output Input or Bidirectional Blocking.

If $\Lambda(Y) \geq \eta_1$, we accept H_0, which is that IP ID acceleration occurred as a result of no inbound blocking. This does not give us a result, as we still must decide between outbound blocking and no blocking. To make this decision, we proceed to the second SHT phase,"Outbound Test," which is discussed subsequently.

A third output of the system is that $\Lambda(Y)$ did not meet either threshold. If there are more trials we restart the algorithm. If we have exhausted our trials, we output the result blockage that of S_i at R_j is undetermined.

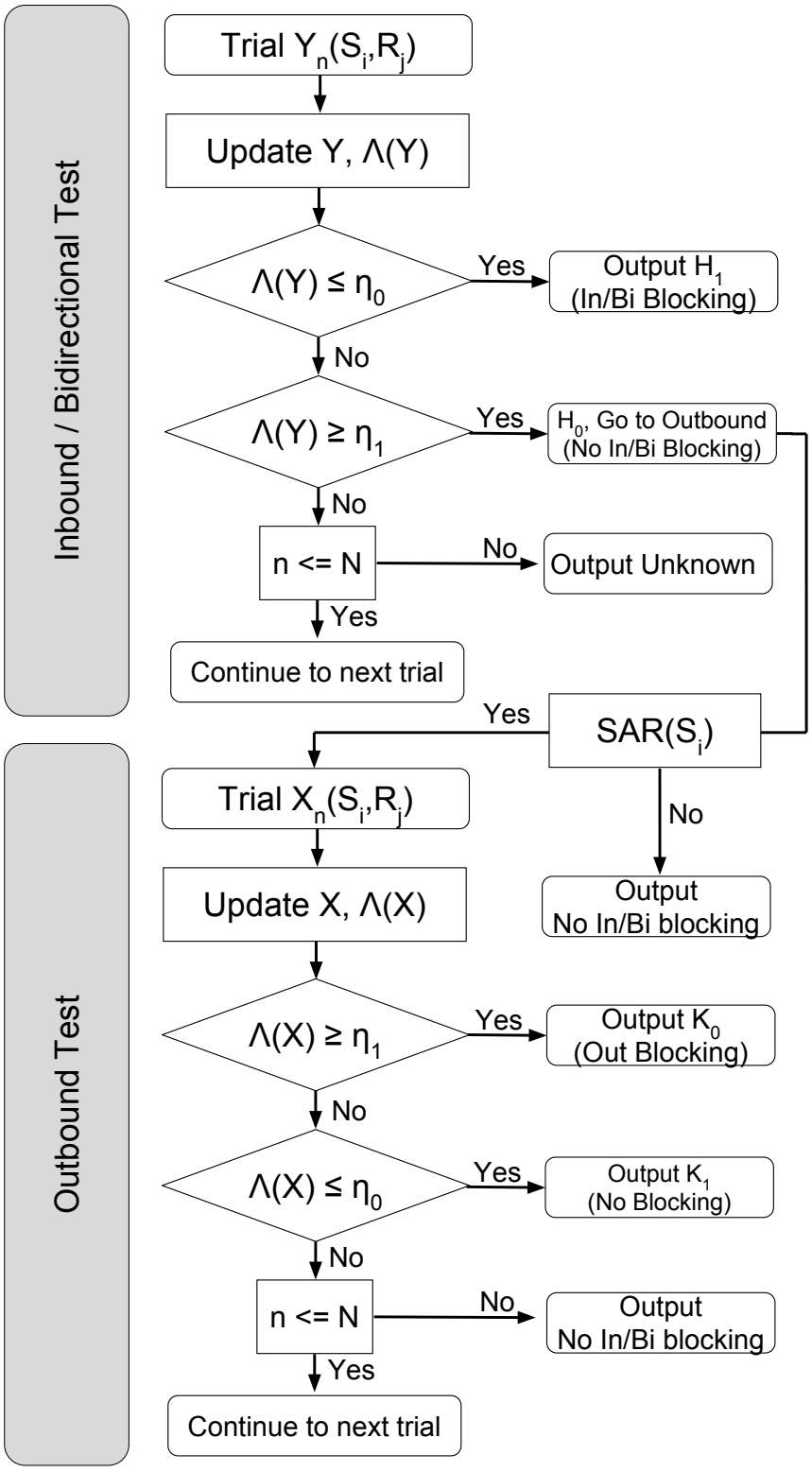


Figure 3.2: Flow chart of our algorithm to identify both inbound and outbound blocking using a series of sequential hypothesis tests. Detailed descriptions of the notation and terminology are given in Section 3.3.

Outbound blocking detection with SHT. Given IP ID acceleration at the reflector, we must distinguish outbound-only blocking from a lack of blocking whatsoever. To do so, we develop a key new insight that relies on a secondary IP ID acceleration that should occur due to subsequent SYN-ACK retries by the site.

To determine a site's eligibility for outbound blocking detection, we must identify whether it retries SYN-ACKs, and that the retries have reliable timing. Section 3.4 discusses these criteria further. We abstract this behavior as a function SAR(S_i) (for SYN-ACK Retry) that indicates whether a site is suitable for outbound blocking detection. We define $X_n(S_i, R_j)$ such that:

$$X_n(S_i, R_j) = \begin{cases} 0 & \text{if no IP ID accel. during SAR} \\ 1 & \text{if IP ID accel. during SAR} \end{cases}$$

We now formulate two new hypotheses, K_0 such that outbound blocking is occurring (IP ID acceleration occurs during the SAR time window), and K_1 such that there is no connection blocking (IP ID acceleration does not occur during the SAR window). From this:

$$\Pr[X_n = 0|K_0] = \theta_0, \quad \Pr[X_n = 1|K_0] = 1 - \theta_0$$
$$\Pr[X_n = 0|K_1] = \theta_1, \quad \Pr[X_n = 1|K_1] = 1 - \theta_1$$

In this construction $1 - \theta_0$ is the measurable probability of observing IP ID acceleration during injection, and θ_1 is the measurable prior probability of seeing no IP ID acceleration during the SAR window across all of the reflector's measurements. Similar arguments hold as above to why these provide conservative estimations of the prior values. (We also discuss the measurable IP ID acceleration during the SAR window in Section 3.4.) Figure 3.2 shows how this construction is used to label S_i, R_j as either outbound-blocked or not blocked. If the thresholds are not met and there are no more trials, we output that we know S_i is not inbound-blocked, but we do not know the outbound-block status.

Expected number of trials. The SHT construction from Jung et al. also provides a framework for calculating the expected number of trials needed to arrive at a decision for H_0 and H_1. The expected values are defined as:

$$E[N|H_0] = \frac{\alpha \ln \frac{\beta}{\alpha} + (1-\alpha) \ln \frac{1-\beta}{1-\alpha}}{\theta_0 \ln \frac{\theta_1}{\theta_0} + (1-\theta_0) \ln \frac{1-\theta_1}{1-\theta_0}},$$
$$E[N|H_1] = \frac{\beta \ln \frac{\beta}{\alpha} + (1-\beta) \ln \frac{1-\beta}{1-\alpha}}{\theta_1 \ln \frac{\theta_1}{\theta_0} + (1-\theta_1) \ln \frac{1-\theta_1}{1-\theta_0}}. \tag{3.1}$$

where α and β are also bounded by functions of the tolerable false positive and negative rates, discussed subsequently. Similar constructions hold for K_0 and K_1. We investigate the expected number of trials for both inbound and outbound blocking further in Section 3.4.

False positives and negatives. Following the construction from Jung et al., α and β are both tunable parameters which are bounded by our tolerance to both false positives and false negatives. P_F is defined as the false positive probability, and P_D as the detection probability. The complement of P_D, $1 - P_D$ is the probability of false negatives. These values express the probability of a false result for a single SHT experiment. However, for our method, we perform numerous SHT experiments across sites and reflectors. To account for these repeated trials we set both P_F and $1 - P_D = 10^{-5}$. Given that as P_F and $1 - P_D$ decrease, the expected number of trials to reach a decision increases, our selection of a small value negatively impacts our ability to make decisions. This effect is somewhat mitigated by the distance between experimentally observed priors, and is explored in more detail in Section 3.4 and Figure 3.4.

3.4 Augur Implementation and Experiment Data

In this section, we discuss the deployment of our approach to measure connectivity disruptions across the Internet, as well as the setup that we use to validate the detection method from Section 3.3.

Selecting Reflectors and Sites

Reflector selection. To find reflectors that satisfy the criteria from Section 3.3, we created a new ZMap [44] probe module that sends SYN-ACK packets and looks for well-formed RST responses. Our module is now part of the open-source ZMap distribution. Using this module, we scan the entire IPv4 address space on port 80 to identify possible reflectors.

We then perform a second set of probes against this list of candidate reflectors to identify a subset that conforms to the desired IP ID behavior. Our tool runs from the measurement machine and sends ten SYN-ACK packets to port 80 of each host precisely one second apart, recording the IP ID of each RST response. We identify reflectors whose IP ID behaviors satisfy the previously outlined requirements: no IP ID wrapping, variable accelerations observed (indicating our packets do induce perturbations in the IP ID dynamics), and a response to all probes. Because the measurement machine induces packet generation at the reflector at a constant rate, any additional IP ID acceleration must be due to traffic from other connections. We further ensure that the measurement machine receives a response for each probe packet that it sends, ensuring that the reflector is stable and reliable enough to support continuous measurements.

This selection method identifies viable reflectors, those that are responsive and exhibit the desired IP ID behavior. We finally filter the viable reflectors that do not correspond to infrastructure, as described in Section 3.2, which significantly reduces the number of available reflectors, as described in Section 3.4.

Site selection. We begin with a list of sites, some of which are expected to be disrupted by network filtering or censorship from a variety of vantage points. We seed our candidate sites with the Citizen Lab list of potentially censored URLs [29], which we call the *CLBL*. This list contains potentially blocked URLs, broken down by category. To further identify sensitive URLs, we use

Reflector Datasets	Total Reflectors	Num. Countries	Median / Country
All Viable	22,680,577	234	1,667
Ethically Usable	53,130	179	15
Experiment Sample	2,050	179	15

Table 3.1: Summary of our reflector datasets. All viable reflectors are identified across the IPv4 address space. Those ethically usable are routers at least two hops away from traceroute endpoints in the Ark data, and we select a random subset as our experiment set. Note that the number of countries includes dependent territories.

Reflector Dataset	AF	AS	EU	NA	SA	OC	ME
All Viable	55	50	52	39	23	14	20
Ethically Usable	36	47	46	30	14	6	18
Experiment Sample	36	47	46	30	14	6	18

Table 3.2: The distribution of countries (and dependent territories) containing reflectors across continents. Note the continent coverage of our experiment sample is identical to that of the ethically usable dataset, as we sampled at least one ethically usable reflector per country in that dataset. The continent labels are as follows: AF=Africa, AS=Asia, EU=Europe, NA=North America, SA=South America, OC=Oceana/Australia. We also label ME=Middle East, as a region with frequent censorship.

Khattak et al.'s dataset [81] that probed these URLs using the OONI [111] measurement platform looking for active censorship. After filtering the list, we distill the URLs down to domain names and resolve all domains to the corresponding IP addresses using the local recursive DNS resolver on a network in the United States. If a domain name resolves to more than one IP, we randomly select one A record from the answers. To augment this list of sites, we randomly select domains from the Alexa top 10,000 [3]. As with the CLBL, if a host resolves to multiple IPs, we select one at random. Section 3.4 provides a breakdown of the site population. Section 3.4 explains how we dynamically enforce site requirements.

Measurement Dataset

In this section, we describe the characteristics of the dataset that we use for our experiments.

Reflector dataset. The geographic distribution of reflectors illuminates the degree to which we can investigate censorship or connectivity disruption within each country. Table 3.1 summarizes the geographic diversity of our reflector datasets. The Internet-wide ZMap scan found 140 million reachable hosts. Approximately 22.7 million of these demonstrated use of a shared, monotoni-

cally increasing IP ID. Using MaxMind [98] for country-level geolocation, these reflectors were geographically distributed across 234 countries and dependent territories[1] around the world, with a median of 1,667 reflectors per country. This initial dataset provides a massive worldwide set of reflectors to potentially measure, yet many may be home routers, servers, or user machines that we cannot use for experimentation due to ethical considerations.

Merging with the Ark to ensure that the reflectors only contain network infrastructure reduces the 22.7 million potential reflectors to only about 53,000. Despite this significant reduction, the resulting dataset contains reflectors in 179 countries (and dependent territories), with a median of 15 reflectors per country. Table 3.2 gives a breakdown of reflector coverage by continent.

We select a subset of these reflectors as our final experiment dataset, randomly choosing up to 16 reflectors in all 179 countries, yielding 1,947 reflectors (not all countries had 16 infrastructure reflectors). In addition to these reflectors, we added 103 high-reliability (stable, good priors) reflectors primarily from China and the US to ensure good coverage with a stable set of reflectors, resulting in 2,050 reflectors in the final dataset. These reflectors also exhibit widespread AS diversity, with the resulting set of reflectors representing 31,188 ASes. Using the Ark dataset to eliminate reflectors that are not infrastructure endpoints reduces this set to 4,214 ASes, with our final experiment sample comprising 817 ASes.

Site dataset. Merging the CLBL with Khattak et al.'s dataset [81] yields 1,210 distinct IP addresses. We added to this set an additional 1,000 randomly selected sites from the Alexa top 10,000. To this set of sites we also added several known Tor bridges, as discussed in Section 3.5. While this set consists of 2,213 sites, some sites appeared in both the CLBL and Alexa lists. Thus, our site list contains a total of 2,134 unique sites, with a CLBL composition of 56.7%.

Experiment Setup

The selection process above left us able to measure connectivity between 2,134 sites and 2,050 reflectors. We collected connectivity disruption network measurements over 17 days, using the method described in Section 3.3. We call one measurement of a reflector-site pair a *run*, involving IP ID monitoring and one instance of blocking detection. Related, we define an experiment *trial* as the complete measurement of one run for all reflector-site pairs. Over our 17-day window, we collected a total of 207.6 million runs across 47 total trials, meaning we tested each reflector-site pair 47 times.

Each run comprises of a collection of one-second time intervals. For each time interval, we measure the IP ID state of the reflector independent of all other tasks. We begin each run by sending a non-spoofed SYN to the site from the measurement machine. Doing so performs several functions. First, it allows us to ensure that the site is up and responding to SYNs at the time of the measurement. Second, it allows us to precisely measure if the site sends SYN-ACK retries, and to characterize the timing of the retries. We record this behavior for each run and incorporate this

[1]Countries and dependent territories are defined by the ISO 3166-1 alpha-2 codes, which is the granularity of Maxmind's country geolocation.

initial data point into the subsequent SHT analysis. We then wait four seconds before injecting spoofed SYN packets towards the site. The reflector measurements during that window serve as *control* measurements. During the injection window, we inject 10 spoofed SYN packets towards the site.

For each run, we denote the SYN-ACK retry behavior and at what subsequent window we expect SYN-ACK retries to arrive at the reflector, and use this information to identify which window to look for follow-on IP ID acceleration. At the end of the run, we send corresponding RST packets for all SYNs we generated, to induce tear-down of all host state. We then cool down for 1 second before starting a new run. We randomize the order of the sites and reflectors for testing per trial. We test all reflector-site pairs before moving on to a new trial. For reasons discussed earlier, we never involve the same reflector and site in two independent simultaneous measurements between endpoints.

After each run, we ensure that (1) the reflector's IP ID appeared to remain monotonically increasing; (2) no packet loss occurred between the measurement machine and the reflector, and (3) the site is up and responding to SYN packets. Additionally, we ensure that the IP ID does not wrap during either the injection window or the SAR window. We discard the measurements if any of these conditions fails to hold. After these validity checks, our dataset contains 182.5 million runs across 1,960 reflectors and 2,089 sites. The reduction in number of sites and reflectors corresponds to unstable or down hosts. We then apply SHT (Section 3.3) to analyze the reachability between these site-reflector pairs.

Measured Priors and Expectations

A critical piece in the construction of our SHT framework is formulating the prior probabilities for each of our hypotheses. Figure 3.3 shows CDFs of the measured prior probabilities of IP ID acceleration for three different scenarios.

The IP ID acceleration of reflectors matches our intuition, where the acceleration decreasing as frequently as it increases across the dataset. We show this with the "No Injection" CDF, with nearly all reflectors having a probability of IP ID acceleration without injection of less than 0.5. Many reflectors have a probability of acceleration far lower, corresponding to reflectors with low or stable traffic patterns. We then use this per-reflector prior for θ_1 in our SHT construction for detecting inbound blocking. While we could instead estimate the value as 0.5, the expected number of trials depends on the separation between the injection and non-injection priors, so if we are able to use a smaller θ_1 (per reflector), this greatly speeds up detection time.

Figure 3.3 also shows the probability of IP ID acceleration under injection. This value approaches 1 for many reflectors and is above 0.8 more than 90% of reflectors. Noticeably, it is, however, quite low, and even 0 for a handful of reflectors. These correspond to degenerate or broken reflectors that we can easily identify due to their low priors, removing them from our experiment (discussed more in Section 3.4). We use this experimentally measured prior as $1 - \theta_0$ in both of our sequential hypothesis tests. This distribution provides a lower bound for the actual probability of IP ID acceleration, as the experimentally measured value includes inbound blocking (i.e., if some sites experience blocking, those values would *lower* the measured value). Inbound-blocked

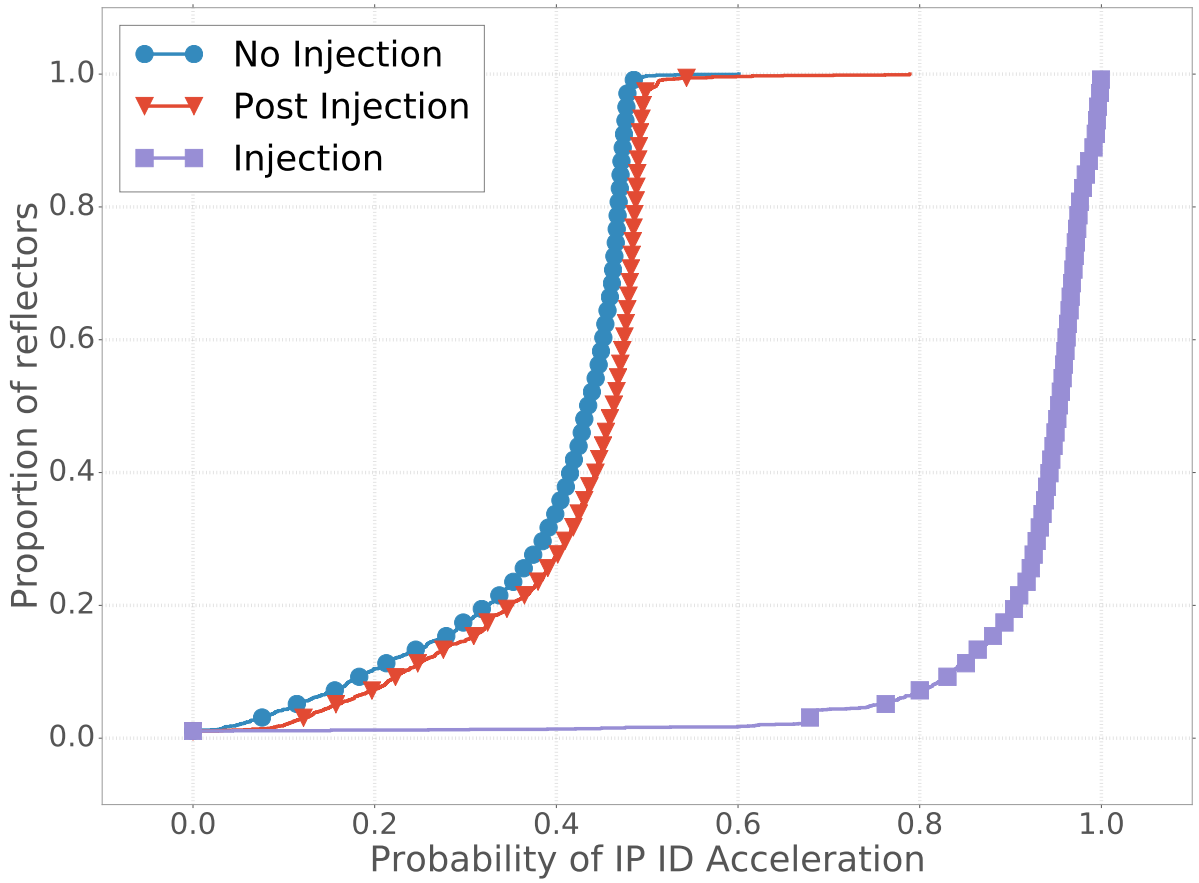


Figure 3.3: CDF of probability of IP ID acceleration per reflector across the experiment.

runs lower the overall probability of acceleration. This still reflects a *conservative* measurement, as a prior closer to control increases the likelihood of false negatives, not false positives.

Lastly, we also measure the probability of IP ID acceleration at the SYN-ACK retry point of each run. We dynamically determine where this falls in each run using the properties the site manifests during that run.[2] As expected, the distribution closely matches the control distribution. The differences in the curve are explained by the dataset containing outbound blocking. Such blocking raises the probability of acceleration at that point, pulling the distribution slightly closer to the injection case. We use this prior as θ_1 during the outbound SHT test.

Once we have computed the priors, we can compute the expected number of trials to reach each of our output states (on a per-reflector basis) using Equation 3.1. Figure 3.4 presents CDFs of these results. More than 90% of reflectors have 40 or fewer expected trials needed to reach one of the states. The remaining reflectors have a large tail and correspond to unstable or degenerate

[2]If a SYN-ACK retry occurs in the window adjacent to injection, we discard that and look for the next retry. If we did not discard that measurement, the retry would correspond to non-acceleration rather than acceleration.

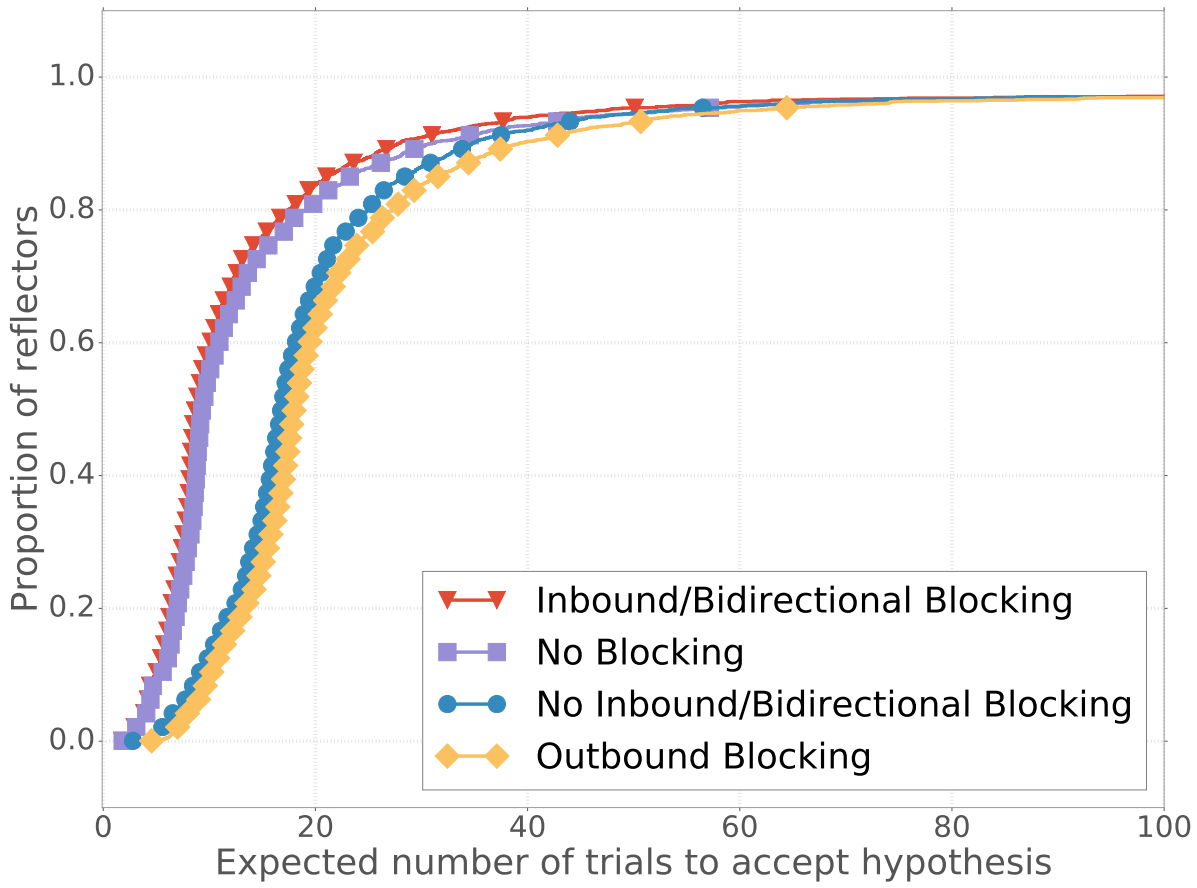


Figure 3.4: CDF of expected number of trials at false positive and negative probability of 10^{-5} to accept one of the four SHT hypothesis outcomes, per reflector. "No Inbound/Bidirectional Blocking" means we passed our first SHT and did not detect inbound blocking, but have not yet attempted to differentiate between no blocking and outbound blocking.

reflectors. We do not need to explicitly remove these reflectors from the dataset, but must refrain from making decisions based on them in some cases.

Identifying and Removing Systematic Effects

Our initial selection of sites did not address some of our site requirements from 3.3, such as network filtering or anycast IP addresses. Failure to identify these sites generates *systematic effects* within our results dataset. Recall that we only wish to filter these sites when necessary. For example, in the case of anycast sites, we can still classify them as inbound-blocked or not blocked, but we cannot detect scenarios where the site is outbound-blocked.

Problematic sites. We identify sites that fail to meet these requirements by conducting a set of experiments with nine geographically diverse vantage points. These hosts reside in cloud service providers and universities, all of which have limited to no network blocking as vantage points. We perform these measurements concurrently with our primary blockage measurements. For each site, we perform two measurements for each vantage point. The diversity of the vantage points enables us to identify these network effects rather than identify censorship or blockage. These tests do not need to be globally complete as the network effects manifest readily.

The first measurement ensures that a vantage point can have bidirectional communication with a site. From a vantage point, we send five SYN packets to a site, evenly distributed over the experiment run (approximately an hour). We monitor for SYN-ACK replies, which demonstrate two-way communication. If a vantage point cannot reliably establish bidirectional communication with a site, we exclude it from our further vantage-point measurements.

In the second measurement, the measurement machine sends a *spoofed* SYN packet to the site with the IP address of a vantage point. Since we previously confirmed the vantage point can communicate with the site, any missing SYN-ACKs or retransmissions are the result of sites not conforming to our requirements, rather than blockage. If the vantage point does not receive a SYN-ACK response from the site, ingress filtering or network origin discrimination may be occurring. If the vantage point does receive a SYN-ACK, it responds with a RST packet. If the vantage point continues to receive multiple SYN-ACKs, the site is not correctly receiving the vantage point's RST packets, suggesting the site host (or its network provider) may be anycast, employing a distributed stateful firewall, or discriminating by traffic origin. We repeat this experiment three times to counter measurement errors introduced by random packet loss. If a vantage point never receives a SYN-ACK, or only ever receives multiple SYN-ACK retries, we conservatively conclude the site exhibits one of the unacceptable network properties from that vantage point. Thus, we disregard its blockage results, *except* if the observed measurement results cannot be a false signal due to the site's properties. For example, if vantages observe only multiple SYN-ACK retries for a site (indicating our measurements with that site may falsely identify outbound blocking), but our measurements detect no blocking or only inbound blocking, we can still consider these results.

We find that this relatively small number of vantage points suffices to characterize sites, as experiment results typically remained consistent across all vantage points. All online sites that we tested were reachable from at least three vantage points, with 98.4% reachable at five or more. This reachability affords us with multiple geographic vantage points to assess each site. For 98.6% of sites, all reachable vantage points consistently assessed the site requirement status, indicating that we can detect site network properties widely from a few geographically distinct locations. This approach is ultimately best effort, as we may fail to detect sites whose behavior is more restricted (e.g., filtering only a few networks).

Through our site assessment measurements, we identified 229 sites as invalid for inbound blocking detection due to ingress filtering or network traffic discrimination. These sites were widely distributed amongst 135 ASes, each of which may employ such filtering individually or may experience filtering occurring at an upstream ISP.

We also flagged 431 sites as invalid for outbound blocking detection as they either lacked a necessary site property (discussed in in Section 3.3) or did not respect RST packets (perhaps

filtering them). To distinguish between the two behaviors, we probed these sites with non-spoofed SYN and RST packets using vantage points, similar to the experiments described earlier in this section. For each site, we sent a SYN packet from a well-connected vantage, and responded with a RST for any received SYN-ACK. If we continued receiving multiple SYN-ACK retries, the site did not respect our RST packets. Otherwise, the site does properly respond to RST packets in the non-spoofing setup, and might be exhibiting an undesirable site property (as listed in Section 3.3) in our spoof-based connectivity disruption experiments. We iterate this measurement three times for robustness against sporadic packet loss, concluding that a site ignores RST packets if any vantage point observes multiple SYN-ACK retransmissions in all trials.

Using this approach, we identified that 64 sites (14.8% of sites invalid for outbound blocking detection) exhibited a non-standard SYN-ACK retransmission behavior, and conclude that the remaining 367 sites (85.2%) are either anycast, deploying stateful firewalls, or discriminating by network origin. These sites were distributed amongst 62 ASes. The majority are known anycast sites, with 75% hosted by CloudFlare and 7% by Fastly, both known anycast networks.

We additionally checked all sites against the Anycast dataset produced by Cicalese et al. [27]. Our technique identified all but 3 IP addresses. We excluded those 3 sites from our results.

Problematic reflectors. A reflector could be subject to filtering practices that differ based on the sender of the traffic, or the port on which the traffic arrives. This systematic effect can manifest as a reflector with significant inbound or outbound blocking. From manual investigation, we identify several reflectors that demonstrate this property independent of spoofed or non-spoofed traffic. In all cases, such reflectors were outliers within their country. To remove these systematic effects, we ignore reflectors in the 99th percentile of blockage for their country. Sites blocked by these reflectors do not show a bias to the CLBL list (discussed more in Section 3.5). This process removed 91 reflectors from our dataset.

3.5 Validation and Analysis

The value of the method we develop ultimately rests on the ability to accurately measure connectivity disruption from a large number of measurement vantage points. Validating its findings presents challenges, as we lack widespread ground truth, presenting a chicken-and-egg scenario. One approach, presented in Sections 3.5 and 3.5, is to analyze the aggregate results produced and confirm they accord with reasonable assumptions about the employment of connectivity disruption. While doing so does not guarantee correctness, it increases confidence in the observations. The other approach is to corroborate our findings against existing ground truth about censored Internet traffic. In Section 3.5, we perform one such analysis, providing a limited degree of more concrete validation.

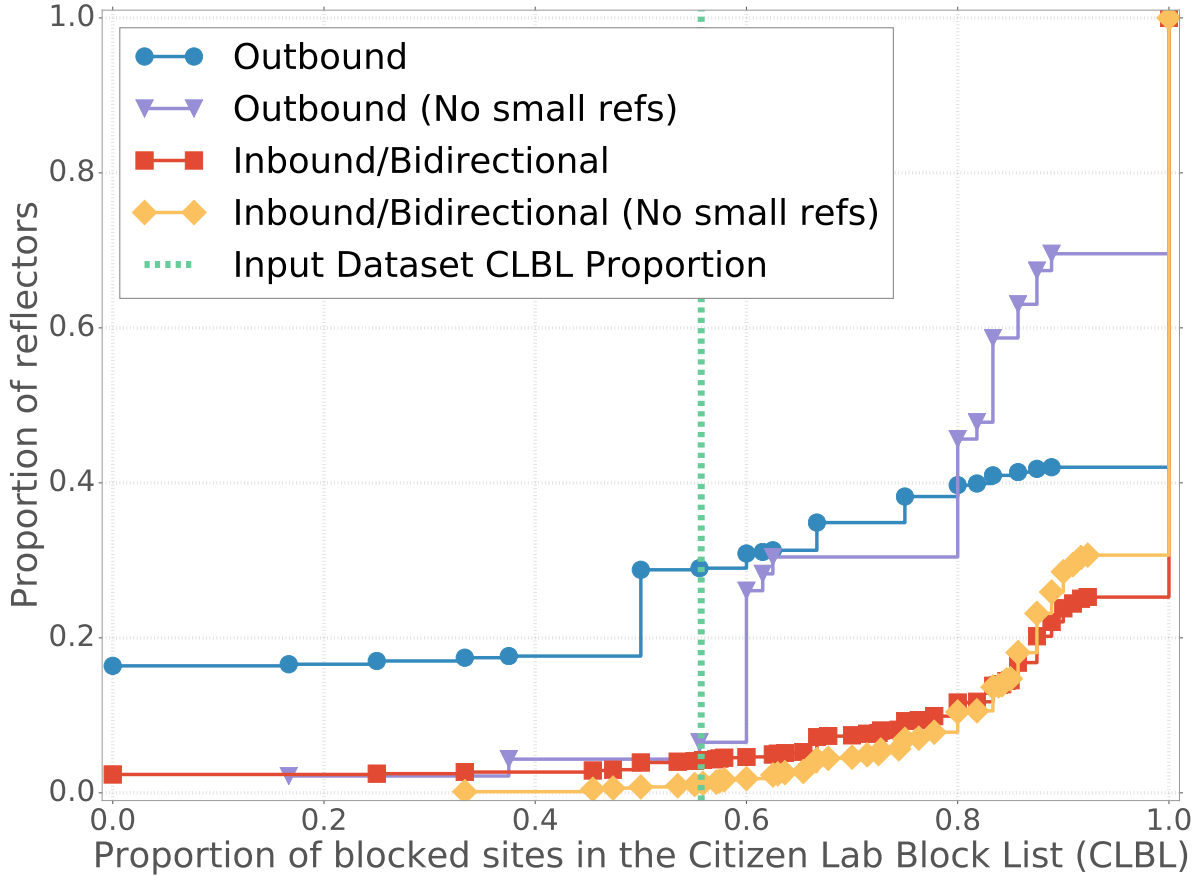


Figure 3.5: Bias of blocked sites towards CLBL sites. CLBL sites consist of 56.7% of our sites, demarcated at the dotted vertical line. To reduce small value effects, we remove reflectors with fewer than 5 blocked websites in curves labeled with "No small refs".

Disruption Bias

Conceptually, one would expect the set of sites disrupted by a network censor to be biased towards sites that are known to be commonly censored. From this notion, we can examine the set of sites blocked by each reflector and ask how that population compares to the input population.

Figure 3.5 shows, in aggregate, the bias of connectivity disruption towards commonly censored websites. 56.7% of websites in the input site dataset are from the CLBL, demarcated in the plot with a vertical dotted line (which we call the CLBL bias line). If the detection we observed was unrelated to censorship, we would expect to find roughly 56.7% of that reflector's blocked sites listed in the CLBL. The results, however, show a considerable bias towards CLBL sites for both inbound and outbound filtering. We see this with the bulk of the graph volume lying to the right of the vertical dotted CLBL bias line. Excluding reflectors with fewer than 5 blocked sites to avoid small number effects, we observe that for 99% of reflectors, more than 56.7% of inbound filtering

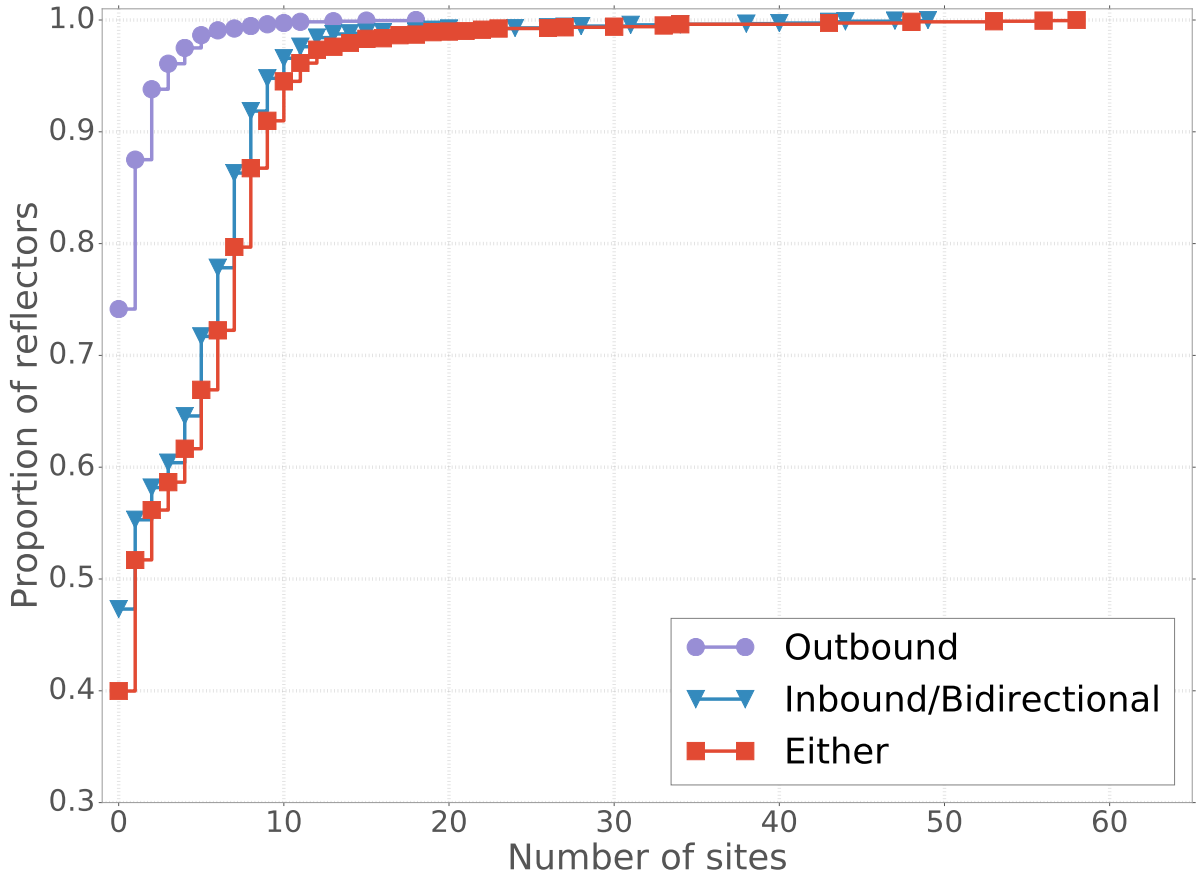


Figure 3.6: CDF of site filtering per reflector, separated by inbound/bidirectional and outbound filtering.

is towards CLBL sites. Similarly, we find 95% of outbound filtering biased towards the CLBL. This observed bias agrees with our prior expectations that we should find CLBL sites more widely censored.

Aggregate Results

Site and reflector results. We first explore the extent of connectivity disruption from both the site and reflector perspective. We might naturally assume that filtering will not manifest ubiquitously. We do not expect to find a site blocked across the majority of reflectors; similarly, we should find most sites not blocked for any given reflector. This should particularly hold since approximately half of our investigated sites come from the Alexa top 10K most visited websites. Although some popular Alexa websites contain potentially sensitive content (e.g., adult or social media sites), many provide rather benign content and are unexpected targets of disruption.

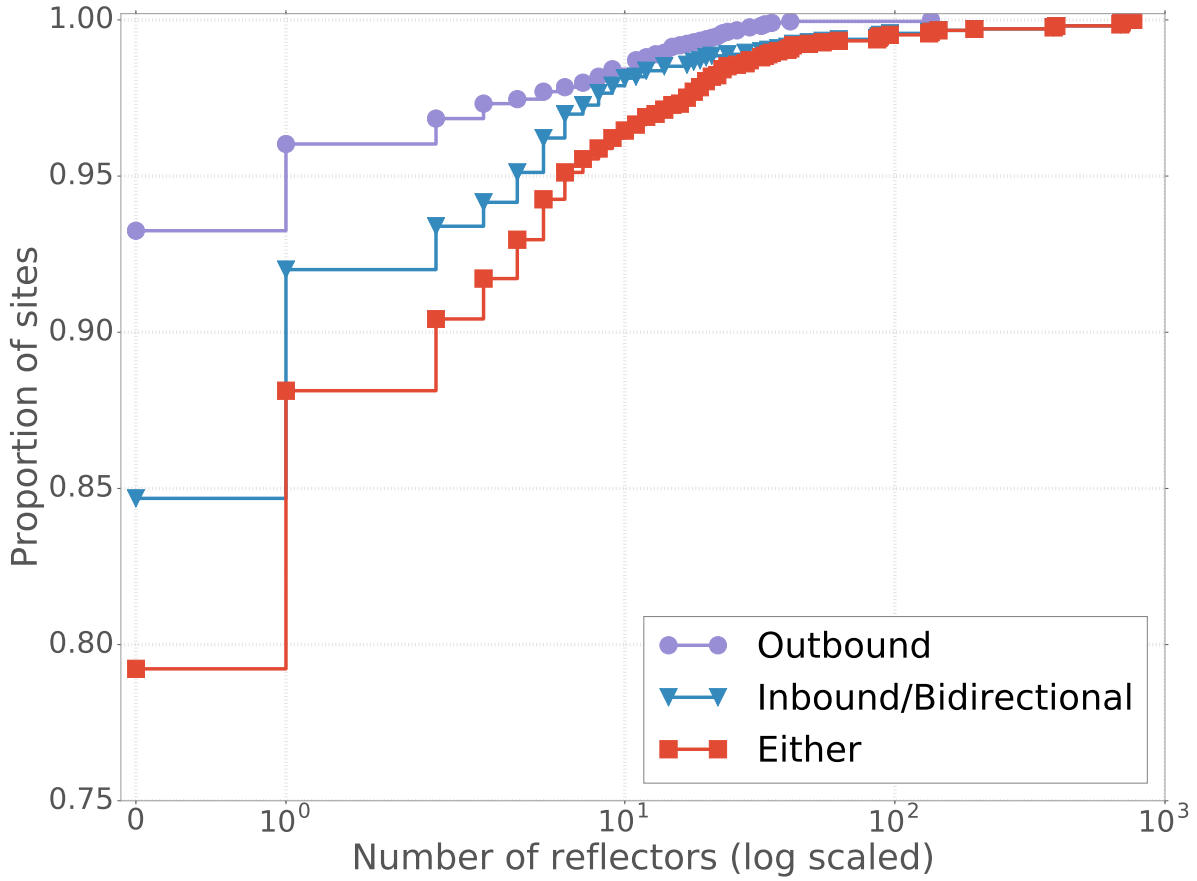


Figure 3.7: CDF of site filtering across reflectors, separated by inbound/bidirectional and outbound filtering. Note the log-scaled x-axis.

We observe the degree of filtering from the reflector perspective in Figure 3.6. Approximately 99% of reflectors encounter connectivity impediments in either direction for 20 or fewer sites, with no reflector blocked for more than 60 sites. This finding concurs with the assumption that site filtering at reflectors is not ubiquitous. On the other hand, connection disruption appears widespread, as 60% of reflectors experience some degree of interference, corroborating anecdotal observations of pervasive censorship.

We find inbound/bidirectional disruption occurs more commonly compared to outbound-only filtering. In total, fewer than 30% of reflectors experience any outbound-only filtering, while over 50% of reflectors have blocked inbound packets from at least one site. This contrast is unsurprising, because bidirectional filtering of a blacklisted IP address is a simple and natural censorship policy; as a result, most results will appear as either inbound or bidirectional filtering.

Figure 3.7 depicts a similar outlook on connectivity disruption from the site viewpoint. We again witness that inbound or bidirectional filtering affects more sites than outbound filtering.

No.	Site	Category	% Refs	% Cnt.
1.	hrcr.org	Human Rights	41.7	83.0
2.	alstrangers.livejournal.com	Militants and Extremists	37.9	78.8
3.	varlamov.ru	Alexa	37.7	78.0
	nordrus-norna.livejournal.com	Hate Speech		
4.	www.stratcom.mil	Foreign Relations & Military	37.5	78.6
5.	www.demonoid.me	Peer-to-Peer File Sharing	21.7	58.5
6.	amateurpages.com	Pornography	21.2	57.9
	voice.yahoo.jajah.com	Voice over Internet Protocol (VOIP)		
	amtrak.com	Alexa		
7.	desishock.net	Peer-to-Peer File Sharing	10.8	32.7
8.	wzo.org.il	Religous Conversion & Commentary	7.9	17.6
9.	Hate Speechit.ru	Hate Speech	7.3	14.5
10.	anonymouse.org	Anonymizers & Censorship Circum.	5.3	16.4

Table 3.3: Summary of the top 10 sites by the percent of reflectors experiencing *inbound* blocking. Rows sharing rank reflect domains that share an IP address. We list a categorization of each website using the CLBL definitions provided. We additionally report the percent of countries (and dependent territories) for which we find a site inbound-blocked by at least one reflector.

Over 15% of sites are inbound-blocked along the path to at least one reflector, while only 7% of sites are ever outbound-blocked. In total, connections to 79% of websites never appear disrupted, and over 99% of sites exhibit inaccessibility by 100 reflectors (5%) or less. As before, these results agree with our expectation that sites are typically not blocked across the bulk of reflectors.

Several sites show extensive filtering, as listed in Tables 3.3 and 3.4. Here, we have determined reflector country-level geolocation using MaxMind [98]. We found six sites inbound-blocked for over 20% of reflectors across at least half the countries (and dependent territories), with the human rights website *hrcr.org* inaccessible by 41.7% of reflectors across 83% of countries. The top 10 inbound-blocked sites correspond closely with anticipated censorship, with 9 found in the Citizen Lab Block List (CLBL). A surprisingly widely blocked Alexa-listed site is *varlamov.ru*, ranked third; in fact, it actually redirects to LiveJournal, a frequent target of censorship [110, 146]. On a related note, the IP address for *amtrak.com* is the sixth most inbound-blocked site—but it is co-located with two CLBL websites, underscoring the potential for collateral damage that IP-based blacklisting can induce.

The top outbound-blocked sites tell a similar tale, although with less pervasive filtering. The most outbound disrupted site is *nsa.gov*, unreachable by 7.4% of reflectors across 23.3% of countries. Given the nature of this site, perhaps the site performs the filtering itself, rather than through reflector-side disruption. All top 10 sites are known frequently blocked websites, listed in the CLBL.

This aggregate analysis of connectivity disruption from both site and reflector perspective accords with our prior understanding that while disruption is not ubiquitous, it may be pervasive. It

No.	Site	Category	% Refs	% Cnt.
1.	nsa.gov	US Government-Run Military Website	7.4	23.3
2.	scientology.org	Minority Faiths	2.2	6.9
3.	goarch.org	Minority Faiths	1.9	4.4
4.	yandex.ru	Freedom of Expression	1.8	3.8
5.	hushmail.com	Email Provider	1.8	4.4
6.	carnegieendowment.org	Political Reform	1.6	4.4
7.	economist.com	Freedom of Expression	1.6	2.5
8.	purevpn.com	Anonymizers & Censorship Circumvention	1.4	1.9
9.	freedominfo.org	Freedom of Expression	1.3	3.1
10.	wix.com	Web Hosting Services	1.3	0.6

Table 3.4: Summary of the top 10 sites by the percent of reflectors experiencing *outbound* blocking. We provide a categorization of each website using the CLBL definitions provided. We additionally report the percent of countries (and dependent territories) for which we find a site inbound-blocked by at least one reflector.

affects a large proportion of reflectors, and can widely suppress access to particular sites. The sites for which our method detects interference closely correspond with known censored websites. This concordance bolsters confidence in the accuracy of our method's results.

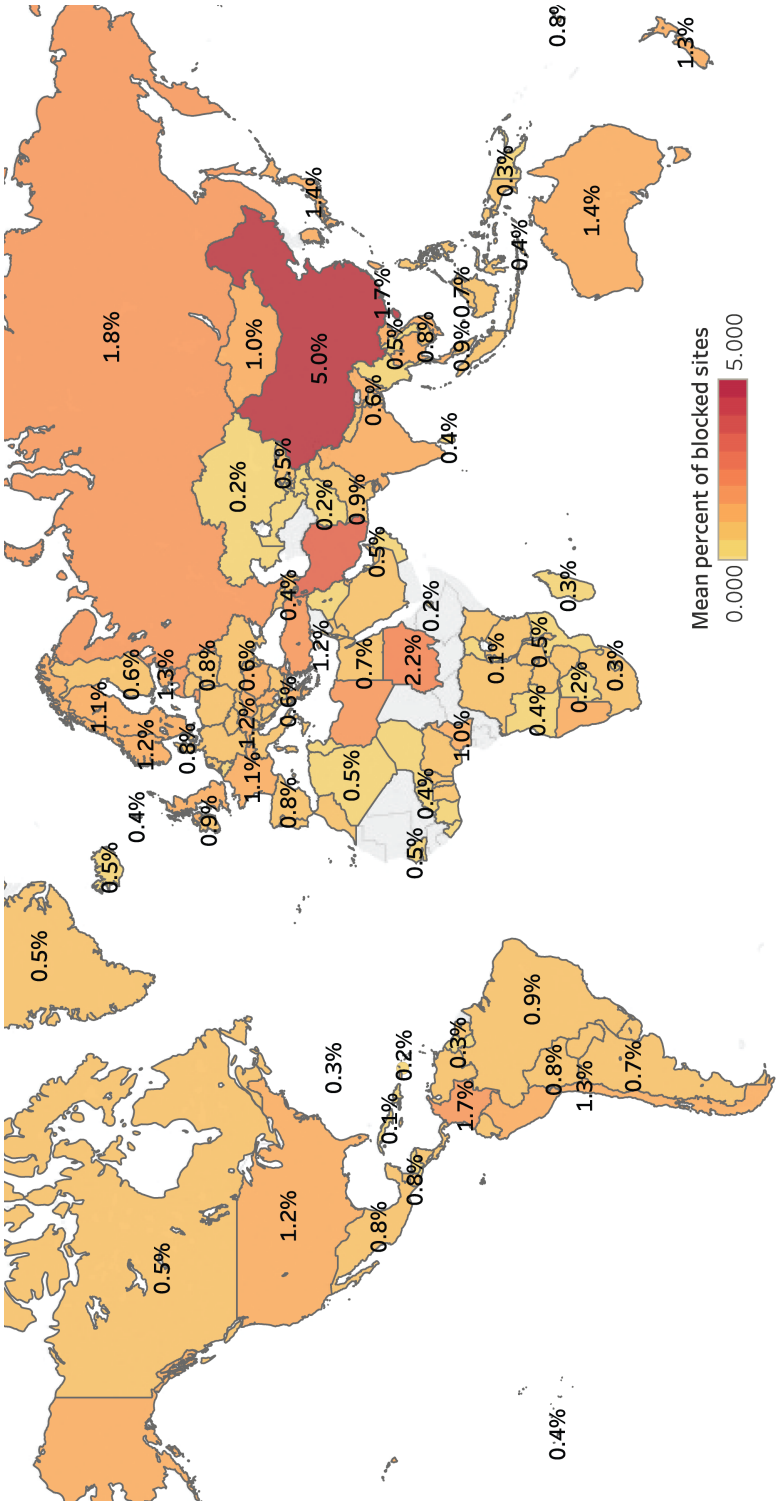


Figure 3.8: Global heat map showing the percentage of sites filtered for any reflector in countries around the world. China experiences the highest average amount of filtering, at 5% of measurable sites filtered by a resolver within the country.

No.	Country	Num. Ref.	Block %	CLBL %	Mean Blocked In/Out	Med. Blocked In/Out	Total Num. Block. In/Out
1.	China	36	5.0	70.9	11.2 / 1.8	1.5 / 0.0	70 / 33
2.	Iran	14	3.4	55.7	10.8 / 1.4	0.0 / 0.0	53 / 17
3.	Sudan	12	2.2	54.3	6.5 / 0.0	1.0 / 0.0	46 / 0
4.	Russia	17	1.8	78.9	4.8 / 1.4	0.0 / 0.0	18 / 20
5.	Latvia	14	1.8	81.6	3.3 / 1.6	2.0 / 0.0	22 / 19
6.	Turkey	15	1.8	83.8	2.1 / 1.5	0.0 / 0.0	23 / 14
7.	Hong Kong	16	1.7	88.9	2.8 / 1.4	0.0 / 0.0	14 / 22
8.	Columbia	16	1.7	85.7	4.2 / 1.2	6.0 / 0.0	17 / 18
9.	Libya	10	1.5	77.4	8.4 / 3.2	9.5 / 3.0	16 / 15
10.	United Kingdom	16	1.4	90.0	3.1 / 0.8	2.0 / 0.0	19 / 11

Table 3.5: Summary of the top 10 countries ranked by the percentage of sites blocked at any reflectors within each country (shown in the "Block %" column). Additionally, we list for each country the number of reflectors within that country, the blockage bias towards CLBL sites, and statistics on inbound versus outbound blockage.

Country-level connectivity disruption. Analysis of aggregate connectivity disruption across countries provides another perspective for validation. Using reflector country geolocation provided by MaxMind [98], Table 3.5 ranks the top 10 countries by percentage of blocked sites across any reflectors in the country. Figure 3.8 portrays this at a global scale, illustrating that some degree of connectivity disruption is experienced by hosts in countries around the world.

We see that many of the most disruptive countries correspond closely with countries known to heavily censor, such as China, Iran, Sudan, Russia, and Turkey [112]. Of the top 10 countries, the OpenNet Initiative [112] has reported Internet censorship of political or social material in every country except Latvia and the United Kingdom.[3] More recently, reports have documented Latvia as heavily censoring gambling websites and political content [6, 133]. Our results appear plausible for the United Kingdom as well, which has a history of filtering streaming and torrent sites [16] and adult content [106].

While we are aggregating at a country granularity, these disruptions may actually be implemented in different ways within a single country. These differences result in non-uniform filtering policies, as has been observed with the Great Firewall of China [49, 151] and UK adult content filtering [106]. In Figure 3.9, we plot the variation in the number of sites blocked for reflectors within each country. We remove countries without any site filtering. We observe that for most countries, there exists some variation in the disruption experienced by reflectors within a country, suggesting that interference indeed often differs across networks even within a country. The extent of this behavior is widespread and highlights the importance of connectivity measurements from

[3] We list Hong Kong separately from China, although traffic from Hong Kong may traverse Chinese networks and experience disruption.

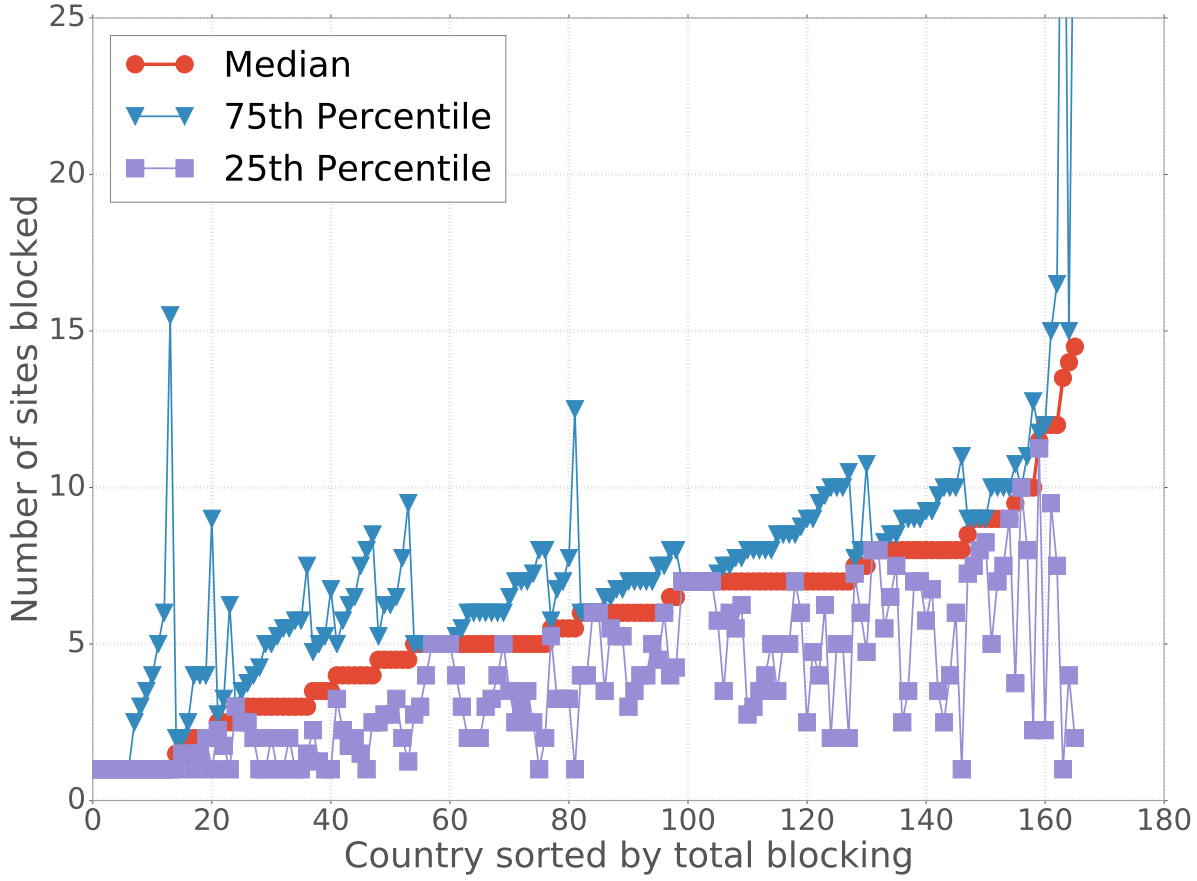


Figure 3.9: Plot of the variations in site filtering experienced by reflectors within countries. We elide countries without any disruption.

many vantage points, since findings may differ across nearby networks and geolocations.

Tor Bridge Case Study

In the previous section, we analyzed our method's results in aggregate, finding them in line with reasonable assumptions and existing reports of Internet censorship. Here, we use several known Tor bridges as a case study providing an additional (though limited) check of correctness. This validation increases confidence in our method, as we are able to replicate previous findings with regards to which sites experience blocking, the country of censorship, and the directional nature of disruption.

Our set of sites contains three Tor Obfuscation4 (obfs4) Bridges open on port 80, for which we have some ground truth on their censorship. A prior study [52] tested all three bridges from vantage points in the U.S., China, and Iran, over a five-month period. The first two bridges (TB1

and TB2) were included in the Tor Browser releases. Fifield and Tsai detected that only China frequently *inbound*-blocked these, albeit inconsistently, likely due to the federated nature of the Great Firewall of China. The third bridge (TB3) had been only privately distributed, and remained unblocked throughout the study.

Our findings are consistent with this ground truth. Both TB1 and TB2 experienced inbound filtering in China only, while connectivity to TB3 was never disrupted. Of the 36 reflectors in China, we detected inbound filtering of TB1 for 8 reflectors, no filtering for 8 reflectors, and inconclusive evidence for the remaining 20 (due to lack of a statistically significant signal during our hypothesis testing). For TB2, 9 reflectors were inbound-blocked, 11 were unblocked, and 16 were undecided. TB3, expected to be unblocked, was accessible by 22 reflectors, with the remaining 14 undetermined. These findings accord with prior results regarding the distributed and disparate nature of Chinese Tor filtering.

3.6 Discussion

In this section, we discuss various aspects concerning the coverage, granularity, and accuracy of the current measurements.

Coverage limitations. Ethical considerations when performing our measurements restricted the reflectors from which we measure to a set of hosts that we can confidently conclude represent Internet infrastructure in the interior of the network. Recall that we do so by measuring the Internet topology and only using reflectors at least two traceroute hops into the network. This approach drastically reduces the number of hosts that we can use as reflectors. In the future, more exhaustive techniques to identify Internet infrastructure could increase the set of IP addresses that we might use as reflectors.

Evasion Augur relies on the injection of spoofed SYN-ACK packets. A natural evasion mechanism could use a stateful firewall to drop SYN-ACKs that do not correspond to a previously sent SYN. Implementing such firewalls at scale poses significant challenges. Large networks frequently have multiple transit links resulting in asymmetric routing; SYN packets may traverse a different path than the SYN-ACKs. The censor would need to coordinate state across these links. Any errors in state management would lead to blocking benign connections, resulting in collateral damage.

Alternatively, censors could switch to allowing through TCP control packets and only disrupting *data* packets. Such an approach might complicate the censor's own monitoring of their blocking efforts as it runs counter to assumptions commonly made by diagnostic tools. Similarly, it may introduce management burdens because it does not accord with common forms of packet filtering.

Ambiguity in location and granularity of filtering. The current measurements only indicate whether packets became filtered somewhere along the end-to-end path between a reflector and a site; they do not indicate the *location* where that filtering might take place. As a result, our

techniques cannot disambiguate the scenario where a remote site blocks access from all reflectors in an entire region from the scenario where an in-country censor filters traffic along that path. For example, financial and commerce sites may block access from entire countries if they have no customers in those regions.

Additionally, the current measurements only employ TCP packets using port 80. Thus, they do not disambiguate filtering of IP addresses versus filtering of only port 80 traffic associated with that IP address. An extension of our system might perform follow-up measurements on different ports to determine whether filtering applies across all ports. On a related note, our techniques only measure TCP/IP-based filtering; future work may involve correlating the measurements that we observe with tools that measure global filtering at other layers or applications (e.g., HTTP, DNS).

Other sources of inaccuracy. Existing IP geolocation tools have known inaccuracies [70], particularly for Internet infrastructure (i.e., IP addresses that do not represent end hosts). As a result, some of our results may not reflect precise characterizations of country-level filtering. As IP geolocation techniques improve, particularly for IP addresses that correspond to Internet infrastructure, we can develop more confidence in the country-level characterizations from Section 3.5. Additionally, various network mechanisms, including anycast, rerouting, traffic shaping, and transient network failures, may make it difficult to disambiguate overt filtering actions from more benign network management practices. Some of these effects may even operate dynamically: for example, network firewalls may observe our probes over time, come to view them as an attack, and begin to block our probes; in this case, our own measurements may give rise to filtering, rendering it difficult to disambiguate reactive filtering of our measurements from on-path filtering between a site and reflector, particularly since the latter may also change over time.

3.7 Augur Summary

Despite the pervasive practice of Internet censorship, obtaining widespread, continuous measurements from a diversity of vantage points has proved elusive; most studies of censorship to-date have been limited both in scale (i.e., concerning only a limited number of vantage points) and in time (i.e., covering only a short time span, with no baseline measurements). The lack of comprehensive measurements about Internet censorship stems from the difficulty of recruiting vantage points across a wide range of countries, regions, and ISPs, as most previous techniques for measuring Internet censorship have required some type of network presence in the network being monitored.

In this chapter, we tackled this problem with a fundamentally different type of approach: instead of relying on in-country monitoring points for which we have no direct access, we exploit recent advances in TCP/IP side-channel measurement techniques to collect measurements between pairs of endpoints that we do not control. This ability to conduct measurements from "third-party" vantage points that we control allows us to continuously monitor many more paths than was previously possible. Previous work introduced the high-level concept of these third-party side-channel measurements; in this work, we transition the concept to practice through a working system that

abides by ethical norms and produces sound measurements in the presence of the measurement artifacts and noise that inevitably manifest in real-world deployments.

The continuous, widespread measurements that we can collect with these techniques can ultimately complement anecdotes, news reports, and policy briefings to ensure that we can back future assessments of Internet filtering with sound, comprehensive data. Part of this transition to practice involves further developing the system that we have developed to facilitate ongoing operation, including automating the validation of the measurements that we collect. In Chapter 4 we expand on and continue this work, by exploring DNS-based filtering, globally.

Chapter 4

Iris: Global Measurement of DNS Manipulation

4.1 Introduction

Organizations may implement censorship at many layers of the Internet protocol stack; they might, for example, block traffic based on IP address, manipulate DNS responses, or they might block individual web requests based on keywords. Prior work has developed techniques to continuously measure widespread manipulation at the transport [47, 116, Chapter 3] and HTTP [128] layers, yet a significant gap remains in our understanding of global information control concerning the manipulation of the Internet's Domain Name System (DNS). Towards this goal, in this chapter we develop and deploy a method and system to detect, measure, and characterize the manipulation of DNS responses in countries across the entire world.

Developing a technique to *accurately* detect DNS manipulation poses major challenges. Although previous work has studied inconsistent or otherwise anomalous DNS responses [75, 93], these methods have focused mainly on identifying DNS responses that could reflect a variety of underlying causes, including misconfigurations. In contrast, our work aims to develop methods for accurately identifying DNS manipulation indicative of an intent to restrict user access to content. To achieve high detection accuracy, we rely on a collection of metrics that we base on the underlying properties of DNS domains, resolutions, and infrastructure.

One set of detection metrics focuses on *consistency*—intuitively, when we query a domain from different locations, the IP addresses contained in DNS responses should reflect hosting from either a common server (i.e., the same IP address) or the same autonomous system. Another set of detection metrics focuses on *independent verifiability*, by comparison to independent information such as the identity in the TLS certificate for the website corresponding to the domain. Each of these metrics naturally lends itself to exceptions: for example, queries from different locations utilizing a content distribution network (CDN) will often receive different IP addresses (and sometimes even different CDNs). However, we can use violations of *all* of the metrics as a strong indicator of DNS manipulation.

In addition to achieving accurate results, another significant design challenge concerns *ethics*. In contrast to systems that explicitly involve volunteers in collecting measurements, methods that send DNS queries through open DNS resolvers deployed across the Internet raise the issue of potentially implicating third parties who did not in fact agree to participate in the measurement. Using "open resolvers" is potentially problematic, as most of these are not actual resolvers but instead DNS forwarders in home routers and other devices [129]. A censor may misattribute requests from these resources as individual citizens attempting to access censored resources.

Reasoning about the risks of implicating individual citizens requires detailed knowledge of how censors in different countries monitor access to censored material and how they penalize such actions. These policies and behaviors may be complex, varying across time, region, individuals involved, and the nature of the censored content; such risks are likely intractable to accurately deduce. To this end, our design takes steps to ensure that, to the extent possible, we only query open DNS resolvers hosted in Internet *infrastructure* (e.g., within Internet service providers or cloud hosting providers), in an attempt to eliminate any use of resolvers or forwarders in the home networks of individual users. This step reduces the set of DNS resolvers that we can use for our measurements from tens of millions to only a few thousand. However, we find that the resulting coverage still suffices to achieve a global view of DNS manipulation, and—importantly—in a safer way than previous studies that exploit open DNS resolvers.

Our work makes the following contributions. First, we design, implement, and deploy Iris, a scalable, ethical system for measuring DNS manipulation. Second, we develop analysis metrics for disambiguating natural variation in DNS responses for a domain from nefarious manipulation. Third, we perform a global measurement study that highlights the heterogeneity of DNS manipulation, across countries, resolvers, and domains. We find that manipulation varies across DNS resolvers even within a single country.

This chapter is based on work that appeared in the USENIX Security Symposium [119] and USENIX ;login:, [120].

4.2 Method

In this section we describe Iris, a scalable, lightweight system to detect DNS manipulation. We begin by scoping the problem space, identifying the capabilities and limitations of various measurement building blocks, and stating our assumptions about the threat model. We explain the process by which we select (1) which domain names to measure, and (2) the vantage points to measure them from, taking into consideration questions of ethics and scalability. We then describe, given a set of measurement vantage points and DNS domain names, how we characterize the results of our measurements and use them to draw conclusions about whether DNS manipulation is taking place, based on either the *consistency* or the *independent verifiability* of the responses that we receive. Next, we consider our technical approach in light of existing ethical norms and guidelines, and explain how various design decisions help us adhere to those principles as much as possible. Finally, we discuss the implicit and technical limitations of Iris.

Overview

We aim to identify DNS manipulation, which we define as the instance of a DNS response both (1) having attributes (e.g., IP addresses, autonomous systems, web content) that are not consistent with respect to a well-defined control set; and (2) returning information that is demonstrably incorrect when compared against independent information sources (e.g., TLS certificates).

Approach Detecting DNS manipulation is conceptually simple: At a high-level, the idea entails performing DNS queries through geographically distributed DNS resolvers and analyzing the responses for activity that suggests that the responses for a DNS domain might be manipulated. Despite its apparent simplicity, however, realizing a system to scalably collect DNS data and analyze it for manipulation poses both ethical and technical challenges. The ethical challenges concern selecting DNS resolvers that do not implicate innocent citizens, as well as ensuring that Iris does not induce undue load on the DNS resolution infrastructure; Section 4.2 explains the ethical guidelines we use to reason about design choices. Section 4.2 describes how Iris selects a "safe" set of open DNS resolvers; The technical challenges center around developing sound methods for detecting manipulation, which we describe in Section 4.2 and Section 4.2.

Identifying DNS names to query Iris queries a list of sensitive URLs compiled by Citizen Lab [29]. We call this list the Citizen Lab Block List (CLBL). This list of URLs is compiled by experts based on known censorship around the world, divided by category. We distill the URLs down to domain names and use this list as the basis of our dataset. We then supplement this list by adding additional domain names selected at random from the Alexa Top 10,000 [3]. These additional domain names help address geographic or content biases in the the CLBL while not drastically increasing the total number of queries.

Assumptions and focus First, Iris aims to identify widespread manipulation at the scale of Internet service providers and countries. We cannot identify manipulation that is targeted at specific individuals or populations or manipulation activities that exploit high-value resources such as valid but stolen certificates. Second, we focus on manipulation tactics that do not rely on stealth; we assume that adversaries will use DNS resolvers to manipulate the responses to DNS queries. We assume that adversaries do not return IP addresses that are incorrect but within the same IP prefix as a correct answer [9, 11, 108]. Finally, when attributing DNS manipulation to a particular country or dependent territory, we rely on the country information available from Censys [43] supplemented with MaxMind's [98] dataset to map a resolver to a specific country (or dependent territory).

Ethics

Our primary ethical concern is minimizing the risks associated with issuing DNS queries via open resolvers for potentially censored or manipulated domains, since these DNS queries could implicate innocent users. A second concern is the query load that Augur induces on authoritative name-

servers. With these concerns in mind, we use the ethical guidelines of the Belmont Report [14] and Menlo Report [41] to frame our discussion.

From the principles of *respect for persons and* beneficence (Section 2.3), we limit all of our measurements to resolvers we are reasonably certain are part of the Internet infrastructure. We also note that issuing DNS queries through tens of millions of resolvers has rapidly diminishing returns, and that using only open resolvers that we can determine are unlikely to correspond to individual users greatly reduces the risk to any individual without dramatically reducing the benefits of our experiment. We note that our consideration of ethics in this regard is a significant departure from previous work that has issued queries through open DNS resolver infrastructure but has not considered ethics [93].

From the principle of *justice* (Section 2.3) we envision that the beneficiaries of the kinds of measurements that we collect using Iris will be those the bear the potential risk. Even in the event that some entity in a country that hosts an open DNS resolver might bear some risk as a result of the measurements we conduct, we envision that those same entities may ultimately benefit from the research, policy-making, and tool development that Iris facilitates.

The additional guideline of *respect for law and public interest* (Section 2.3) helps us reason about the externalities that our DNS queries create by increasing DNS query load on the nameservers and resolvers. In keeping with this principle, we rate-limit our DNS queries for each DNS domain based on peak and near peak query rate limitations and billing.

Open DNS Resolvers

To obtain a wide range of measurement vantage points, we use *open DNS resolvers* deployed around the world; such resolvers will resolve queries for any client.

Measurement using open DNS resolvers is an ethically complex issue. Previous work has identified tens of millions of these resolvers around the world [93]. Given their prevalence and global diversity, open resolvers are a compelling resource, providing researchers with considerable volume and reach. Unfortunately, open resolvers also pose a risk not only to the Internet but to individual users.

Open resolvers can be the result of configuration errors, frequently on end-user devices such as home routers [93]. Using these devices for measurement can incur monetary cost, and if the measurement involves sensitive content or hosts, can expose the owner to harm. Furthermore, open resolvers are also a common tool in various online attacks such as Distributed Denial-of-Service (DDoS) amplification attacks [94]. Despite efforts to reduce both the prevalence of open resolvers and their potential impact [113], they remain commonplace.

Due to these and the ethics considerations that we discussed in Section 4.2, we restrict the set of open resolvers that we use to the few thousand resolvers that we are reasonably certain are part of the Internet infrastructure (e.g., belonging to Internet service providers, online cloud hosting providers), as opposed to attributable to any single individual. Figure 4.1 illustrates the process by which Iris finds safe open DNS resolvers. We now explain this process in more detail. Conceptually, the process comprises two steps: (1) scanning the Internet for open DNS resolvers;

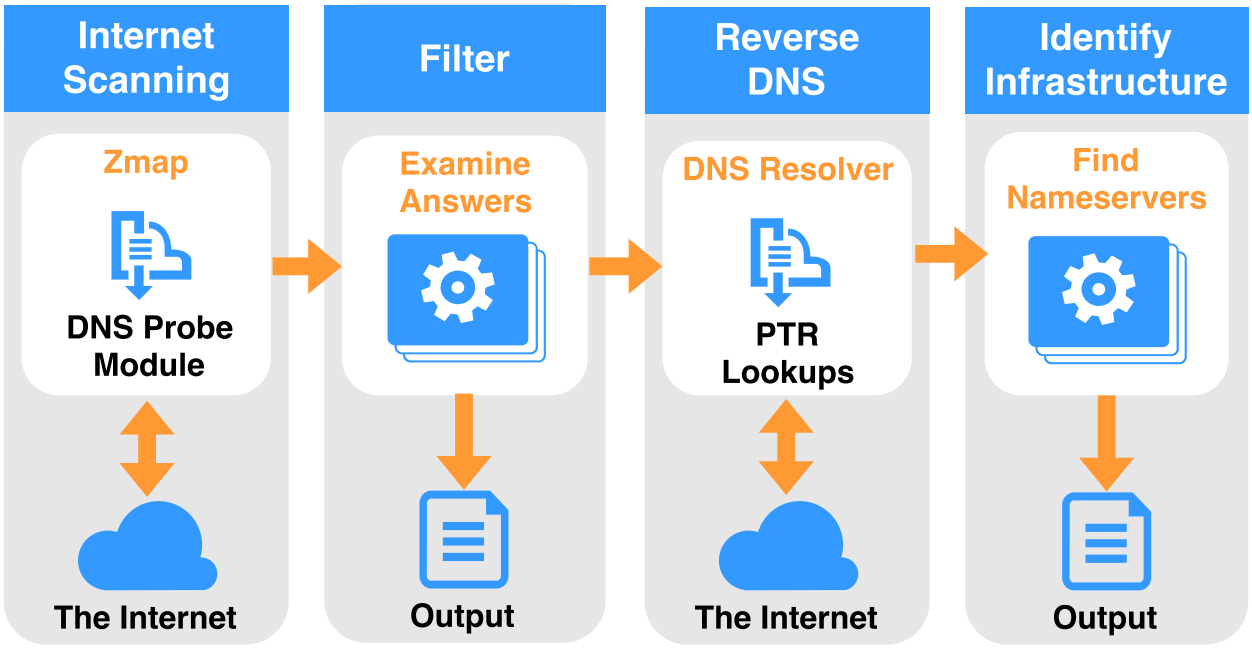


Figure 4.1: Overview of Iris's DNS resolver identification and selection pipeline. Iris begins with a global scan of the entire IPv4 address space, followed by reverse DNS PTR lookups for all open resolvers, and finally filtering resolvers to only include DNS infrastructure.

or (2) pruning the list of open DNS resolvers that we identify to limit the resolvers to a set that we can reasonably attribute to Internet infrastructure.

By using DNS resolvers we do not control, we cannot differentiate between country-wide or state-mandated censorship and localized manipulation (e.g., captive portals, malware [93]) at individual resolvers. Therefore we must aggregate and analyze results at ISP or country scale.

Step 1: Scanning the Internet's IPv4 space for open DNS resolvers Scanning the IPv4 address space provides us with a global perspective on all open resolvers. To do so, we developed an extension to the ZMap [44] network scanner to enable Internet-wide DNS resolutions[1]. This module queries port 53 of all IPv4 addresses with a recursive DNS A record query. We use a purpose-registered domain name we control for these queries to ensure there is a known correct answer. We conduct measurements and scans from IP addresses having a `PTR` record identifying the machine as a "research scanner." These IP addresses also host a webpage identifying our academic institution and offering the ability to opt-out of scans. From these scans, we select all IP addresses that return the correct answer to this query and classify them as open resolvers. In Section 4.3, we explore the population of open DNS resolvers that we use for our study.

[1]Our extension has been accepted into the open source project and the results of our scans are available as part of the Censys [43] system.

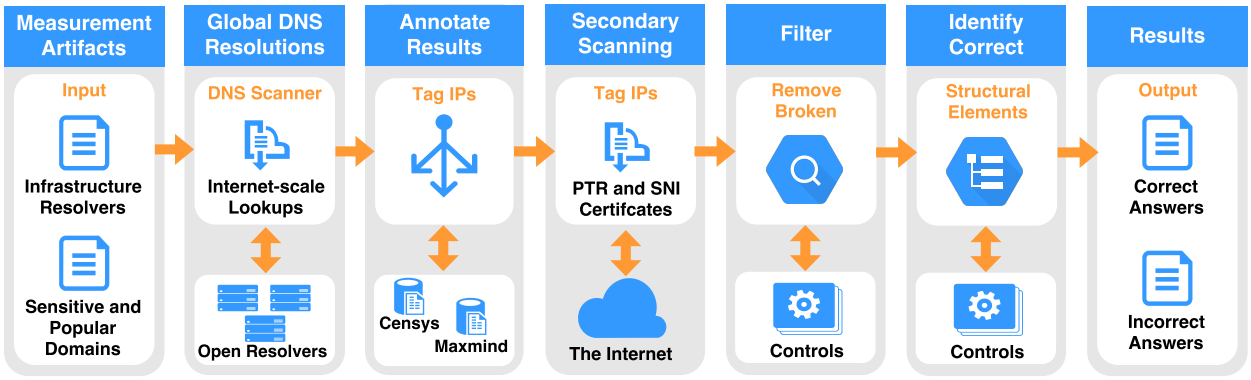


Figure 4.2: Overview of DNS resolution, annotation, filtering, and classification. Iris inputs a set of domains and DNS resolvers and outputs results indicating manipulated DNS responses.

Step 2: Identifying Infrastructure DNS Resolvers Given a list of all open DNS resolvers on the Internet, we prune this list to include only DNS resolvers that can likely be attributed to Internet infrastructure. To do so, we aim to identify open DNS resolvers that appear to be authoritative nameservers for a given DNS domain. Iris performs reverse DNS `PTR` lookups for all open resolvers and retains only the resolvers that have a valid `PTR` record beginning with the subdomain `ns[0-9]+` or `nameserver[0-9]*`. This filtering step reduces the number of usable open resolvers—from millions to thousands—yet even the remaining set of open DNS resolvers provides broad country- and network-level coverage (characterized further in Section 4.3).

Using `PTR` records to identify infrastructure can have both *false negatives* and *false positives*. Not all infrastructure resolvers will have a valid `PTR` record, nor will they all be authoritative nameservers. These false negatives limit the scope and scale of our measurement, but are necessary to reduce risk. Similarly, if a user operated their own authoritative nameserver on their home IP or if a `PTR` record matched our naming criteria but was not authoritative, our method would identify that IP as infrastructure (false positives).

Performing the Measurements

Given a list of DNS domain names to query and a global set of open DNS resolvers from which we can issue queries, we need a mechanism that issues queries for these domains to the set of resolvers that we have at our disposal. Figure 4.2 shows an overview of the measurement process. At a high level, Iris resolves each DNS domain using the global vantage points afforded by the open DNS resolvers, annotates the response IP addresses with information from both outside datasets as well as additional active probing, and uses *consistency* and *independent verifiability* metrics to identify manipulated responses. The rest of this section outlines this measurement process in detail, while Section 4.2 describes how we use the results of these measurements to ultimately identify manipulation.

Step 1: Performing global DNS queries Iris takes as input a list of suitable open DNS resolvers, as well as the combined CLBL and Alexa domain names. In addition to the DNS domains that we are interested in testing, we include 3 DNS domains that are under our control to help us compute our consistency metrics when identifying manipulation.

Querying tens of thousands of domains across tens of thousands of resolvers required the development of a new DNS query tool, because no existing DNS measurement tool supports this scale. We implemented this tool in Go [57]. The tool takes as input a set of domains and resolvers, and coordinates random querying of each domain across each resolver. The tool supports a variety of query types, multiple of which can be specified per run, including `A`, `AAAA`, `MX`, and `ANY`. For each (domain, resolver) pair, the tool crafts a recursive DNS request and sends it to the resolver. The recursive query requests that the resolver resolve the domain and return the ultimate answer, logging all responses, including timeouts. The tool follows the set of responses to resolve each domain to an IP address. For example, if a resolver returns a `CNAME`, the tool then queries the resolver for resolution of that `CNAME`.

To ensure resolvers are not overloaded, the tool includes a configurable rate-limit. For our experiments, we limited queries to resolvers to an upper bound of 5 per second. In practice, this rate tends to be much lower due to network latency in both reaching the resolver, as well as the time it takes the resolver to perform the recursive response. To cope with specific resolvers that are unstable or timeout frequently, the tool provides a configurable failure threshold that halts a specific resolver's set of measurements should too many queries fail.

To ensure the domains we query are not overloaded, the tool randomizes the order of domains and limits the number of resolvers queried in parallel such that in the worst case no domain experiences more than 1 query per second, in expectation.

Step 2: Annotating DNS responses with auxiliary information Our analysis ultimately relies on characterizing both the *consistency* and *independent verifiability* of the DNS responses that we receive. To enable this classification we first must gather additional details about the IP addresses that are returned in each of the DNS responses. Iris annotates each IP address returned in the set of DNS responses with additional information about each IP address's geolocation, autonomous system (AS), port 80 `HTTP` responses, and port 443 `HTTPS` X.509 certificates. We rely on the Censys [43] dataset for this auxiliary information; Censys provides daily snapshots of this information. This dataset does not contain every IP address; for example, the dataset does not include IP addresses that have no open ports, or adversaries may intentionally return IP addresses that return error pages or are otherwise unresponsive. In these cases, we annotate all IP addresses in our dataset with AS and geolocation information from the Maxmind service [98].

Additional PTR and TLS scanning For each IP address, we perform a DNS `PTR` lookup to assist with some of our subsequent consistency characterization (a process we detail in Section 4.2). Another complication in the annotation exercise relates to the fact that in practice a single IP address might host many websites via `HTTP` or `HTTPS` (i.e., virtual hosting). As a result, when Censys retrieves certificates via port 443 (`HTTPS`) across the entire IPv4 address space, the certifi-

cate that Censys retrieves might differ from the certificate that the server would return in response to a query via TLS's Server Name Indication (SNI) extension. Such a discrepancy might lead Iris to mischaracterize virtual hosting as DNS inconsistency. To mitigate this effect, for each resulting IP address we perform an additional active `HTTPS` connection using SNI, specifying the name originally queried. We annotate all responses with this information, which we use for answer classification (examined further in Section 4.4).

Identifying DNS Manipulation

To determine whether a DNS response is manipulated, Iris relies on two types of metrics: *consistency* metrics and *independent verifiability* metrics. We say that a response is correct if it satisfies *any* consistency or independent verifiable metric; otherwise, we classify the response as *manipulated*. In this section, we outline each class of metrics as well as the specific features we develop to classify answers. The rest of this section defines these metrics; Section 4.4 explores the efficacy of each of them.

Consistency

Access to a domain should have some form of *consistency*, even when accessed from various global vantage points. This consistency may take the form of network properties, infrastructure attributes, or even content. We leverage these attributes, both in relation to control data as well as across the dataset itself, to classify DNS responses.

Consistency Baseline: Control Domains and Resolvers Central to our notion of consistency is having a set of geographically diverse resolvers we control that are (presumably) not subject to manipulation. These controls give us a set of high-confidence correct answers we can use to identify consistency across a range of IP address properties. Geographic diversity helps ensure that area-specific deployments do not cause false-positives. For example, several domains in our dataset use different content distribution network (CDN) hosting infrastructure outside North America. As part of our measurements we insert domain names we control, with known correct answers. We use these domains to ensure a resolver reliably returns unmanipulated results for non-sensitive content (e.g., not a captive portal).

For each domain name, we create a set of consistency metrics by taking the union of each metric across all of our control resolvers. For example, if Control A returns the answer `192.168.0.10` and `192.168.0.11` and Control B returns `192.168.0.12`, we create a set of consistent IP set of (`192.168.0.10, 192.168.0.11, 192.168.0.12`). We say the answer is "correct" (i.e., not manipulated) if, for each metric, the answer is a non-empty subset of the controls. Returning to our IP example, if a global resolver returns the answer (`192.168.0.10, 192.168.0.12`), it is identified as correct. When a request returns multiple records, we check all records and consider the reply good if any response passes the appropriate tests.

Additionally, unmanipulated passive DNS [10] data collected simultaneously with our experiments across a geographically diverse set of countries could enhance (or replace) our consistency metrics. Unfortunately we are not aware of such a dataset being available publicly.

IP Address The simplest consistency metric is the IP address or IP addresses that a DNS response contains.

Autonomous System / Organization In the case of geographically distributed sites and services, such as those hosted on CDNs, a single domain name may return different IP addresses as part of normal operation. To attempt to account for these discrepancies, we also check whether different IP addresses for a domain map to the same AS we see when issuing queries for the domain name through our control resolvers. Because a single AS may have multiple AS numbers (ASNs), we consider two IP addresses with either the same ASN or AS *organization name* as being from the same AS. Although many responses will exhibit AS consistency even if individual IP addresses differ, even domains whose queries are not manipulated will sometimes return inconsistent AS-level and organizational information as well. This inconsistency is especially common for large service providers whose infrastructure spans multiple regions and continents and is often the result of acquisitions. To account for these inconsistencies, we need additional consistency metrics at higher layers of the protocol stack (specifically HTTP and HTTPS), described next.

HTTP Content If an IP address is running a webserver on port 80, we include a hash of the content returned as an additional consistency metric. These content hashes come from a port 80 IP address Censys crawl. This metric effectively identifies sites with limited dynamic content. As discussed in Section 4.4, this metric is also useful in identifying sites with dynamic content but shared infrastructure. For example, as these hashes are based on `HTTP GET` fetches using an IP address as the `Host` in the header, this fetch uniquely fingerprints and categorizes CDN failures or default host pages. In another example, much of Google's web hosting infrastructure will return the byte-wise identical redirection page to `http://www.google.com/` for `HTTP GET`s without a valid Google host header. These identical pages allow us to identify Google resolutions as correct even for IP addresses acting as a Point-of-Presence.

HTTPS Certificate We label a response as correct if the hash of the `HTTPS` certificate presented upon connection matches that of an IP returned via our controls. Note this is not an independent verifiability metric, as the certificates may or may not be trusted, and may not even be correct for the domain.

PTRs for CDNs From our control data, we classify domains as hosted on particular CDNs based on `PTR`, AS, and certificate information. We consider a non-control response as consistent if the `PTR` record for that response points to the same CDN.

Independent Verifiability

In addition to consistency metrics, we also define a set of metrics that we can independently verify using external data sources, such as the HTTPS certificate infrastructure. We describe these methods below.

HTTPS Certificate We consider a DNS response to be correct, independent of controls, if the IP address presents a valid, browser-trusted certificate for the correct domain name when queried without SNI. We further extend this metric to allow for common configuration errors, such as returning certificates for `*.example.com` when requesting `example.com`.

HTTPS Certificate with SNI We add an additional metric that checks whether the certificate returned from our follow-up SNI-enabled scans returns a valid, browser-trusted certificate for the correct IP address.

Limitations

To facilitate global coverage in our measurements, our method has limitations that impact our scope and limit our results.

Localized Manipulation Since Iris relies entirely on open infrastructure resolvers that we do not control, in regions with few resolvers, we cannot differentiate between localized manipulation by the resolver's operator and ISP or country-wide manipulation. Analysis of incorrect results focusing on consistency across ISP or country, or examination of webpage content, could aid in identifying localized manipulation.

Domain Bias From our set of infrastructure resolvers, we measure manipulation of the CLBL and a subset of Alexa top sites. Although the CLBL is a community-based effort to identify sensitive content globally, by its very nature it is not *complete*. URLs and domains are missing, and sensitive content may change faster than the list is updated. Similarly, the list may exhibit geographic *bias* based on the language of the project and who contributes to it. This bias could affect the relative volume and scope of manipulation that Iris can detect.

Evasion Although we focus on manipulation at ISP or country scale, an active adversary can still attempt to evade our measurements. Upstream resolvers could use EDNS Client Subnet [32] to only manipulate results for certain target IP ranges, or ISP resolvers could choose to manipulate only their own customers. Country-wide firewalls that perform injection could identify our scanning IP addresses and either not inject results or block our communication entirely. An adversary could also exploit our consistency metrics and inject incorrect IP addresses within the same AS as the targets.

Resolver Datasets	Total Resolvers	Number Countries	Median / Country
All Usable	4,197,543	232	659.5
Ethically Usable	6,564	157	6.0
Experiment Set	6,020	151	6.0

Table 4.1: DNS resolver datasets. We identify all correctly functioning open resolvers are across the IPv4 address space. The experiment set consists of resolvers that passed additional functional tests beyond our basic scan. Note that the number of countries includes dependent territories.

Geolocation Error We rely on Censys [43] and Maxmind [98] for geolocation and AS labeling of infrastructure resolvers to perform country or ISP-level aggregation. Incorrect labeling would identify country-wide manipulation as incomplete (false negatives), or identify manipulation in countries where it is not present (false positives).

4.3 Dataset

In this section, we characterize the data collected and how we processed it to obtain the results used in our analysis.

Open Resolver Selection

We initially identified a large pool of open DNS resolvers through an Internet-wide ZMap scan using our DNS extension to ZMap in January 2017. In total, 4.2 million open resolvers responded with a correct answer to our scan queries. This number excludes resolvers that replied with valid DNS responses but had either a missing or incorrect IP resolution for our scan's query domain.

The degree to which we can investigate DNS manipulation across various countries depends on the geographic distribution of the selected DNS resolvers. By geolocating this initial set of resolvers using Censys [43] and MaxMind [98], we observed that these resolvers reside in 232 countries and dependent territories[2], with a median of 659 resolvers per country. Due to the ethical considerations we outlined in Section 4.2, we restrict this set of resolvers to 6,564 infrastructure resolvers, in 157 countries, again with a median of 6 resolvers per country. Finally, we remove unstable or otherwise anomalous resolvers; Section 4.3 describes this process in more detail. This filtering reduces the set of usable resolvers to 6,020 in 151 countries, with a median of 6 resolvers in each. Table 4.1 summarizes the resulting population of resolvers; Table 4.2 shows the breakdown across continents. We also use 4 geographically diverse resolvers for controlled experiments; the 2 Google Public DNS servers [62], a German open resolver hosted on Amazon AWS, and a resolver that we manage at the University of California, Berkeley.

[2]Countries and dependent territories are defined by the ISO 3166-1 alpha-2 codes, the granularity of Maxmind's country geolocation.

Resolver Dataset	AF	AS	EU	NA	OC	SA
All Usable	55	49	52	41	21	14
Ethically Usable	29	42	42	25	8	11
Experiment Set	26	41	41	24	8	11

Table 4.2: Number of countries (and dependent territories) containing usable resolvers by continent. AF=Africa, AS=Asia, EU=Europe, NA=North America, OC=Oceana/Australia, SA=South America.

Response Datasets	Total Responses	Number Resolvers	Number Domains
All Responses	14,539,198	6,564	2,330
After Filtering	13,594,683	6,020	2,303

Table 4.3: DNS response dataset before and after filtering problematic resolvers, domains, and failed queries.

Domain Selection

We investigate DNS manipulation for both domains known to be censored and domains for popular websites. We began with the Citizen Lab Block List (CLBL) [29], consisting of 1,376 sensitive domains. We augment this list with 1,000 domains randomly selected from the Alexa Top 10,000, as well as 3 control domains we manage that should not be manipulated. Due to overlap between the two domain sets, our combined dataset consists of 2,330 domains. We excluded 27 problematic domains that we identified through our data collection process, resulting in our final population of 2,303 domains.

Response Filtering

We issued 14.5 million DNS `A` record queries for our 2,330 pre-filtered domains, across 6,564 infrastructure and control open resolvers during a 2 day period in January 2017. We observed various erroneous behavior that required further filtering. Excluding these degenerate cases reduced our dataset collection to 13.5 million responses across 2,303 domains and 6,020 resolvers, as summarized in Table 4.3. The rest of this section details this filtering process.

Resolvers We detected that 341 resolvers stopped responding to our queries during our experiment. An additional 202 resolvers incorrectly resolved our control domain names, despite previously answering correctly during our Internet-wide scans. The common cause of this behavior was rate limiting, as our Internet-wide scans queried resolvers only once, whereas our experiments necessitated repeated queries. We identified another problematic resolver that exhibited a query

Failure Type	Count	% of Responses
Timeout	140,551	0.97%
Server Fail	107,826	0.74%
Conn Refused	7,823	0.05%
Conn Error	3,686	0.03%
Truncated	3,451	0.02%
NXDOMAIN	1,713	0.01%

Table 4.4: Breakdown of the 265,050 DNS responses that returned a non-success error code.

failure rate above 70% due to aggressive rate limiting. We eliminated these resolvers and their associated query responses from our dataset, reducing the number of valid responses by 510K.

Domains Our control DNS resolvers could not resolve 15 domain names. We excluded these and their associated 90K query responses from our dataset. We removed another 12 domains and their 72K corresponding query responses as their DNS resolutions failed an automated sanity check; resolvers across numerous countries provided the same incorrect DNS resolution for each of these domains, and the IP address returned was unique per domain (i.e., not a block page or filtering appliance). We did not expect censors to exhibit this behavior; a single censor is not likely to operate across multiple countries or geographic regions, and manipulations such as block pages that use a single IP address across countries should also be spread across multiple domains. These domains do not support HTTPS, and exhibit geographically specific deployments. With increased geographic diversity of control resolvers or deployment of HTTPS by these sites, our consistency or verifiability metrics would account for these domains.

Queries We filtered another 256K queries that returned failure error codes; 93.7% of all errors were timeouts and server failures. Timeouts denote connections where the resolver did not respond to our query within 15 seconds. Server failures indicate when a resolver could not recursively resolve a domain within its own pre-configured time allotment (10 seconds by default in BIND). Table 4.4 provides a detailed breakdown of error responses.

Returning an NXDOMAIN response code [108], which informs a client that a domain does not exist, is an obvious potential DNS censorship mechanism. Unfortunately, some CDNs return this error in normal operations, presumably due to rate limiting or client configuration settings. We found that the most prevalent NX behavior occurred in the countries of Tonga and Pakistan; both countries exhibited censorship of multiple content types, including adult and LGBT. Previous studies have observed NXDOMAIN blocking in Pakistan [108]. These instances comprise a small percentage of overall NXDOMAIN responses. Given the many non-censorship NXDOMAIN responses and the relative infrequency of their use for censorship, we exclude these from our analysis. Another 72K responses had a `SUCCESS` response code, but contained no IP address in the response. This failure mode frequently coincide with `CNAME` responses that could not

Country	% NXDOMAIN
Tonga	2.93%
Pakistan	0.37%
Bosnia/Herzegovina	0.12%
Isle of Man	0.04%
Cape Verde	0.04%

Table 4.5: The top 5 countries / dependent territories by the percent of queries that responded with NXDOMAIN.

be resolved further. We excluded these queries. Table 4.5 provides a geographic breakdown of NXDOMAIN responses.

After removing problematic resolvers, domains, and failed queries, the dataset comprises of 13,594,683 DNS responses. By applying our *consistency* and *independent verifiability* metrics, we identify 41,778 responses (0.31%) as manipulated, spread across 58 countries (and dependent territories) and 1,408 domains.

4.4 Results

We now evaluate the effectiveness of our DNS manipulation metrics and explore manipulated DNS responses in the context of Internet censorship.

Evaluating Manipulation Metrics

To assess the effectiveness of the *consistency* and *independent verifiability* metrics, we quantify the ability of each metric to identify unmanipulated responses (to exclude from further investigation). Figure 4.3 shows each metric's efficacy. The horizontal axis represents the fraction of responses from a particular resolver that are classified as correct by a given metric. The vertical axis indicates the number of resolvers that exhibit that same fraction of correct responses (again under the given metric). For example, almost 6,000 resolvers had roughly 8% of their responses identified as correct under the "Same CDN" metric. A narrow band indicates that many resolvers exhibit similar fractions of correct responses under that metric (i.e., it is more *stable*). The closer the center mass of a histogram lies to 1.0, the more *effective* its corresponding metric, since a larger fraction of responses are classified as correct (i.e., not manipulation) using that metric.

The AS consistency metric ("Same AS") is the most effective: it classified 90% of the DNS responses as consistent. Similarly, identifying matching IP addresses between responses from our control resolvers and our experiment resolvers flagged about 80% of responses as correct across most resolvers. "Same HTTP Page" is also relatively effective, as many geographically distributed deployments of the same site (such as with Points-of-Presence) have either identical content or infrastructure error characteristics (see Section 4.2). This figure also illustrates the importance of

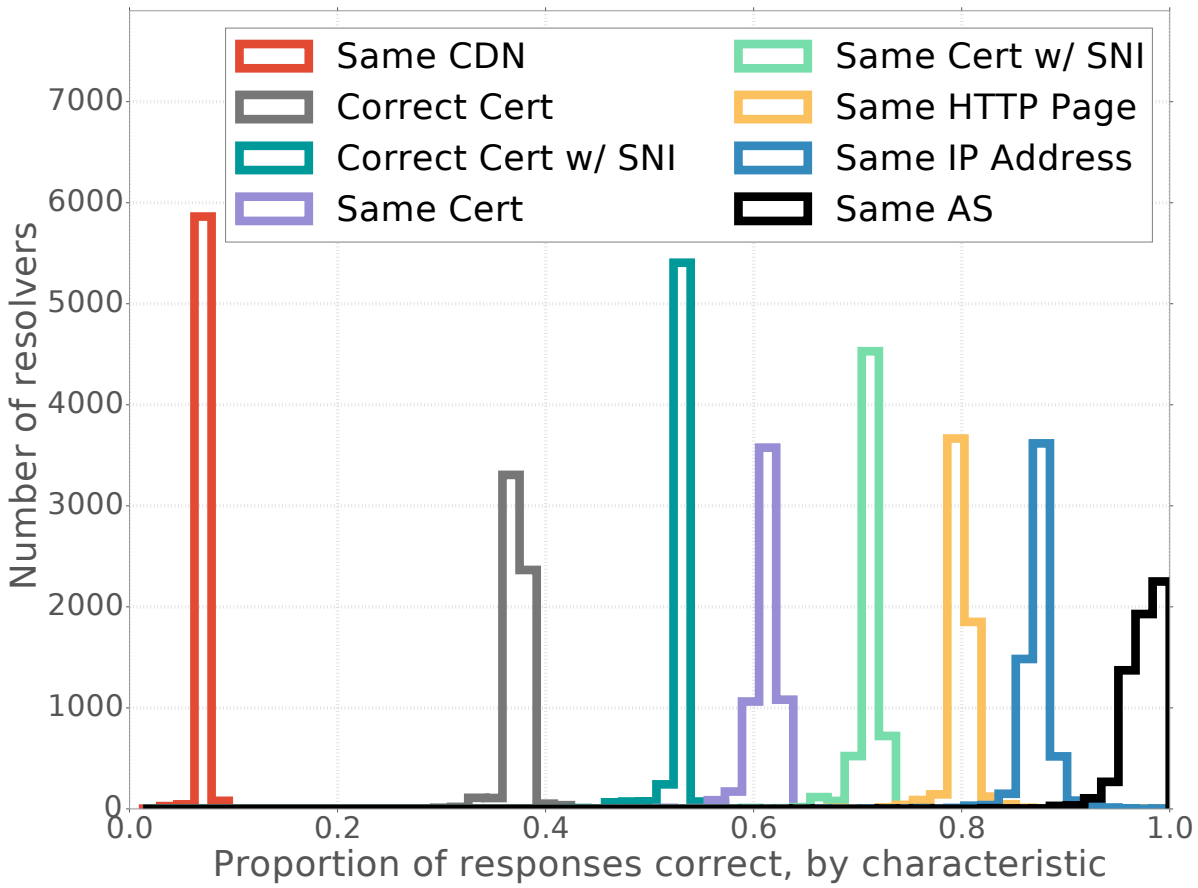


Figure 4.3: The ability of each correctness metric to classify responses as correct. Table is ordered (top to bottom, left to right) by the lines on the graph (left to right).

SNI, increasing the effectiveness of correct and valid HTTPS certificates from 38% to 55%. The same HTTPS certificate ("Same Cert") metric turns out to be more effective than simply having a correct certificate ("Correct Cert"), because so many sites incorrectly deploy HTTPS.

Manipulated DNS Responses

We detect nearly 42,000 manipulated DNS responses; we now investigate the distribution of these responses across resolvers, domains, and countries.

Manipulated responses by resolver Figure 4.4 shows the cumulative fraction of results that return at least a certain fraction of manipulated responses: 88% of resolvers exhibited no manipulation; for 96% of resolvers, we observe manipulation for fewer than 5% of responses. The modes in the CDF highlight differences between resolver subpopulations, which upon further investiga-

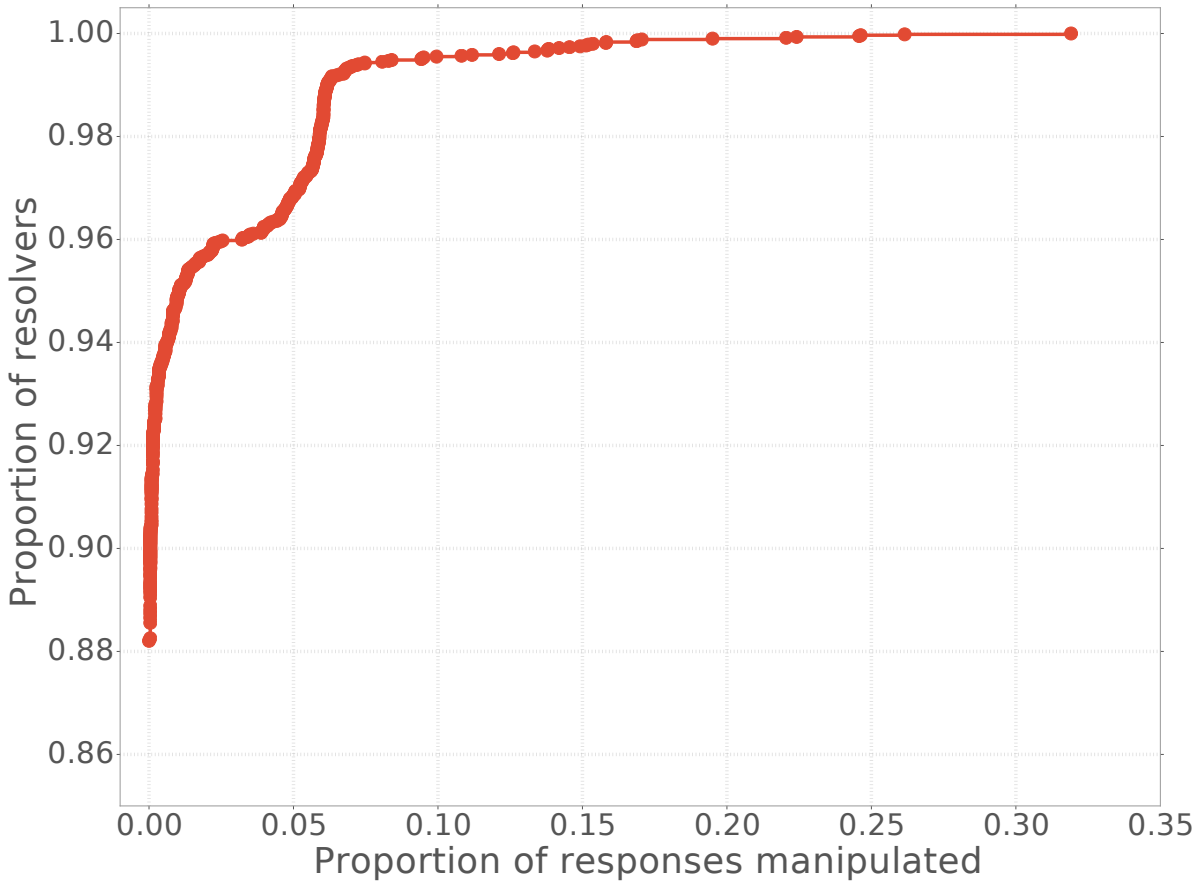


Figure 4.4: The fraction of responses manipulated, per resolver. For 89% of resolvers, we observed no manipulation.

tion we discovered reflected differing manipulation practices across countries. Additionally, 62% of domains are manipulated by at least one resolver, which is expected given that more than half of our selected domains are sensitive sites on the CLBL. We explore these variations in more detail later in this section.

Manipulated responses by country Previous work has observed that some countries deploy nation-wide DNS censorship technology [9]; therefore, we expected to see groups of resolvers in the same country, each manipulating a similar set of domains. Table 4.6 lists the percent of manipulated responses per resolver, aggregated across resolvers in each country. Resolvers in Iran exhibited the highest degree of manipulation, with a median of 6.02% manipulated responses per Iranian resolver; China follows with a median value of 5.22%. These rankings depend on the domains in our domain list, and may merely reflect that the CLBL contained more domains that are censored in these countries.

Country (# Res.)	Median	Mean	Max	Min
Iran (122)	6.02%	5.99%	22.41%	0.00%
China (62)	5.22%	4.59%	8.40%	0.00%
Indonesia (80)	0.63%	2.81%	9.95%	0.00%
Greece (26)	0.28%	0.40%	0.83%	0.00%
Mongolia (6)	0.17%	0.18%	0.36%	0.00%
Iraq (7)	0.09%	1.67%	5.79%	0.00%
Bermuda (2)	0.04%	0.04%	0.09%	0.00%
Kazakhstan (14)	0.04%	0.30%	3.90%	0.00%
Belarus (18)	0.04%	0.07%	0.30%	0.00%

Table 4.6: Top 10 countries by median percent of manipulated responses per resolver. We additionally provide the mean, maximum, and minimum percent for resolvers in each country. The number of resolvers per country is listed with the country name.

The top 10 countries shown in Table 4.6 all have at least one resolver that does not manipulate *any* domains; IP address geolocation inaccuracy may partially explain this surprising finding. For example, uncensored resolvers in Hong Kong may be incorrectly labeled as Chinese. Additionally, for countries that do not directly implement the technical manipulation mechanisms but rather rely on individual ISPs to do so, the actual manifestation of manipulation may vary across ISPs within a single country. Localized manipulation by resolver operators in countries with few resolvers could also influence these results. Section 4.4 investigates these factors further.

Figure 4.5 shows the representation of responses in our dataset by country. For example, the leftmost pair of bars shows that, while less than 5% of all responses in our dataset came from Iranian resolvers, the responses that we received accounted for nearly 40% of manipulated responses in the dataset. Similarly, Chinese resolvers represented 1% of responses in the data but contributed to 15% of the manipulated responses. In contrast, 30% of our DNS responses came from resolvers in the United States, but accounted for only 5% of censored responses.

Table 4.7 shows the breakdown of the top manipulated responses, by the IP address that appears in the manipulated answer. The top two special-purpose (i.e., private) IP addresses appear in the majority of responses within Iran. The third most common response is an OpenDNS (a DNS filtering and security product [28]) blockpage indicating adult content. The fourth most frequent response is an IP address hosting an HTTP error page known to be used in Turkey DNS manipulation [18].

Private and special-purpose IPv4 addresses in manipulated DNS responses Of the roughly 42,000 manipulated DNS responses, 17,806 correspond to special-purpose IPv4 addresses as defined by RFC 6890 [33]; the remaining 23,972 responses corresponded to addresses in the public IP address space. Table 4.8 shows the extent to which countries return private IP addresses in responses, for the top 10 countries ranked by the relative amount of DNS manipulation compared

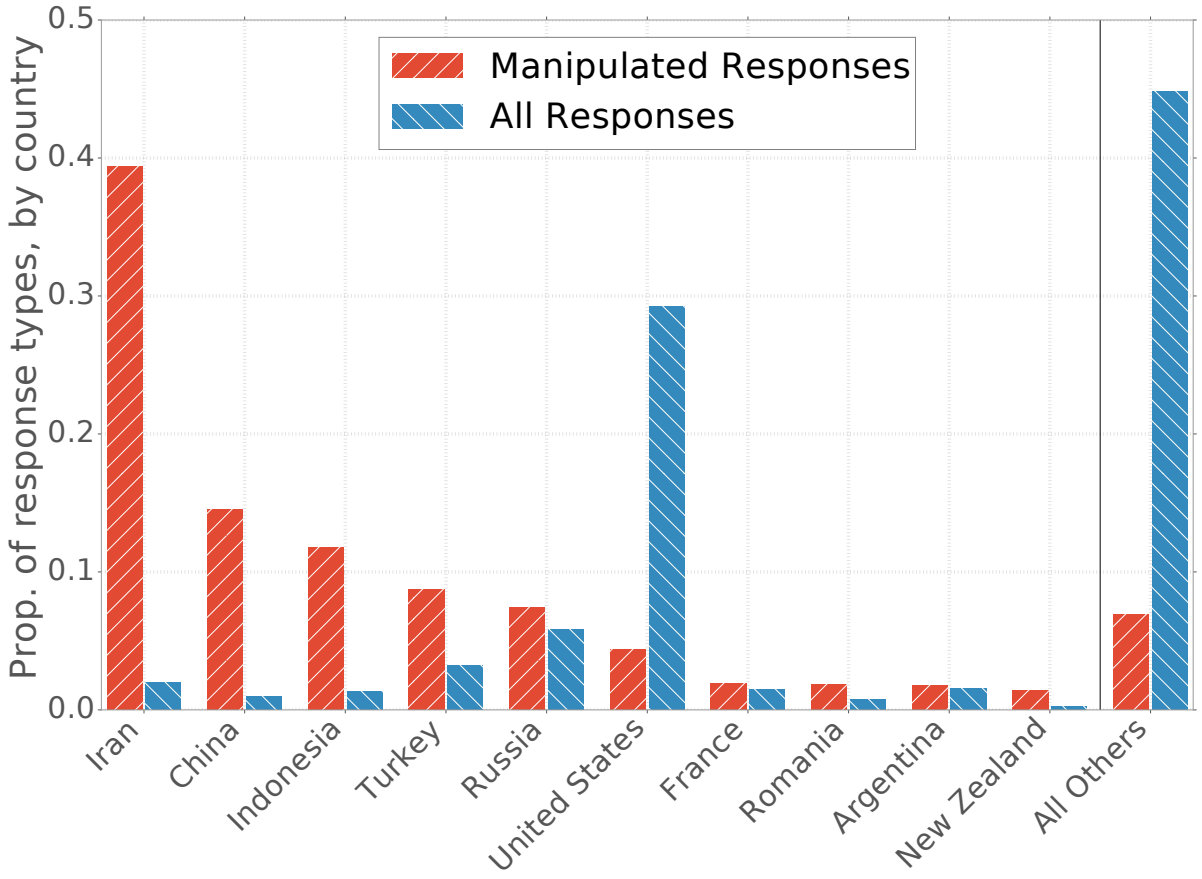


Figure 4.5: The fraction of all responses in our dataset from each country (blue), and the fraction of all manipulated responses in our dataset from the corresponding country (red).

to the total number of results from that country. For example, we observed more manipulated responses from Turkey than Iraq, but Iris used more open DNS resolvers in Turkey, so observed frequencies require normalization. Here, we notice that countries that manipulate DNS tend to either return only special-purpose IP addresses in manipulated responses (as in the case of Iran, Iraq, and Kuwait) or only public IP addresses (China).

Figure 4.6 presents the distribution of observed public IP addresses across manipulated responses in our dataset. The most frequently returned public IP address, an OpenDNS blockpage, constituted almost 15% of all manipulated responses containing public IP addresses. The top ten public IP addresses accounted for nearly 60% of responses.

Many IP answers have been observed in previous studies on Chinese DNS censorship [9, 50]. These addresses are seemingly arbitrary; they host no services, not even a fundamental webpage. The 10 most frequent Chinese responses constituted almost 75% of Chinese responses. The remaining 25% are spread over a long tail of nearly 1,000 seemingly arbitrary non-Chinese IP ad-

Answer	Results	Names	Category
10.10.34.36	12,144	140	Private
10.10.34.34	4,566	776	Private
146.112.61.106	3,495	801	OpenDNS Adult
195.175.254.2	3,137	129	HTTP Error Page
93.46.8.89	1,571	88	China*
118.97.116.27	1,212	155	Safe / Filtering
243.185.187.39	1,167	88	China*
127.0.0.1	876	267	Private
95.53.248.254	566	566	Resolver's Own IP
95.53.248.254	565	565	Resolver's Own IP
8.7.198.45	411	75	China*
202.169.44.80	379	113	Safe / Filtering
212.47.252.200	371	371	Resolver's Own IP
212.47.254.200	370	370	Resolver's Own IP
213.177.28.90	352	22	Gambling Blockpg
208.91.112.55	349	320	Blockpg
180.131.146.7	312	145	Safe / Filtering
203.98.7.65	303	78	China*
202.182.48.245	302	100	Adult Blockpg
93.158.134.250	258	86	Safe / Filtering

Table 4.7: Most common manipulated responses by volume, with manual classification for public, non-resolver IP addresses. The category "China*" are IP addresses previously observed by Farnan et al. in 2016 [50].

Country (# Res.)	% Incor.	% Pub.
Iran (122)	6.02%	0.01%
China (62)	4.52%	99.46%
Indonesia (80)	2.74%	95.08%
Iraq (7)	1.68%	1.49%
New Zealand (16)	1.59%	100.00%
Turkey (192)	0.84%	99.81%
Romania (45)	0.77%	100.00%
Kuwait (10)	0.61%	0.00%
Greece (26)	0.41%	100.00%
Cyprus (5)	0.40%	100.00%

Table 4.8: Percent of public IP addresses in manipulated responses, by country. Countries are sorted by overall frequency of manipulation.

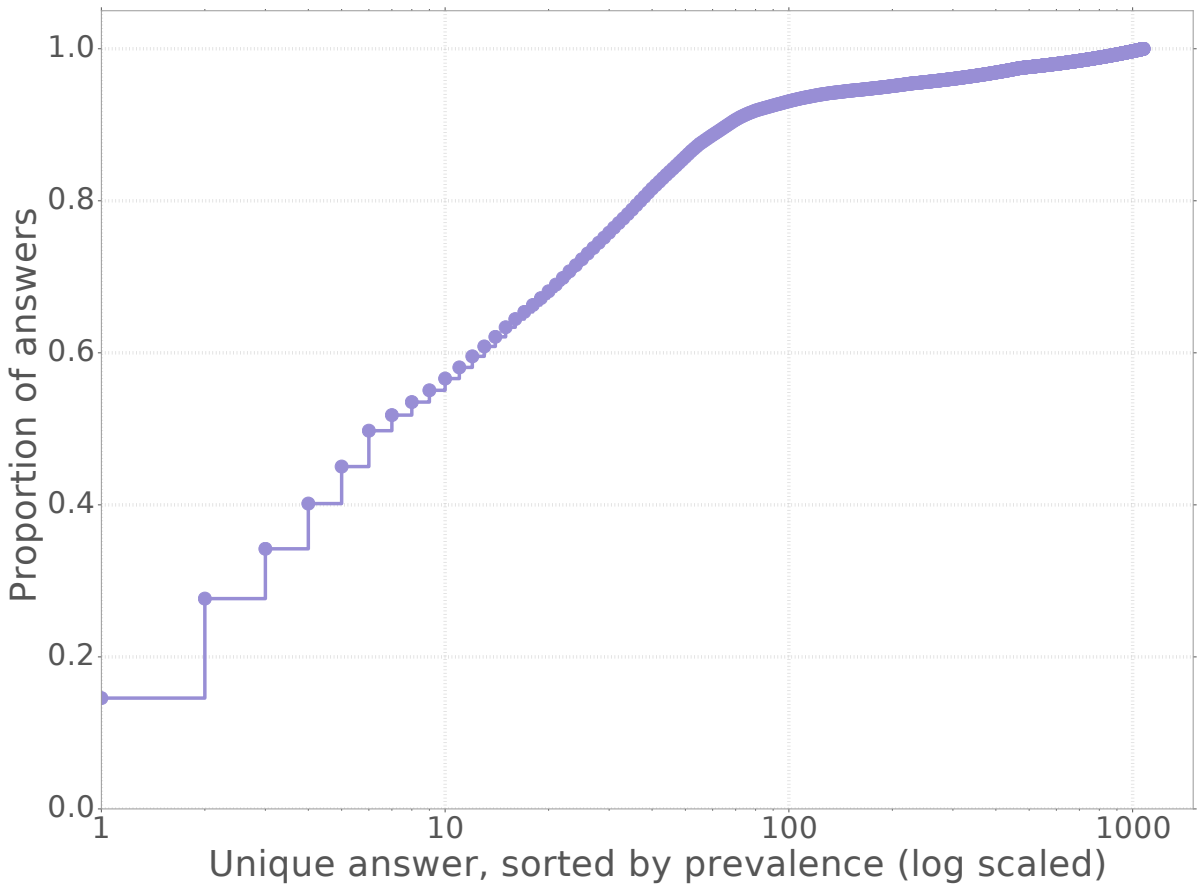


Figure 4.6: Manipulated but public IP addresses in our dataset. The horizontal axis is sorted by the most common IP.

dresses.

Manipulation Within Countries

Figure 4.7 shows the DNS manipulation of each domain by the fraction of resolvers *within* a country, for the 10 countries with the most normalized amount of manipulation. Each point represents a domain; the vertical axis represents the fraction of resolvers in that country that manipulate it. Shading shows the density of points for that part of the distribution. The plot reveals several interesting phenomena. One group of domains is manipulated by about 80% of resolvers in Iran, and another group is manipulated by fewer than 10% of resolvers. This second group of domains is manipulated by a smaller fraction of resolvers, also returning non-public IP addresses. These effects are consistent with previously noted blackholing employed by DNS manipulation infrastructure [11]; this phenomenon deserves further investigation.

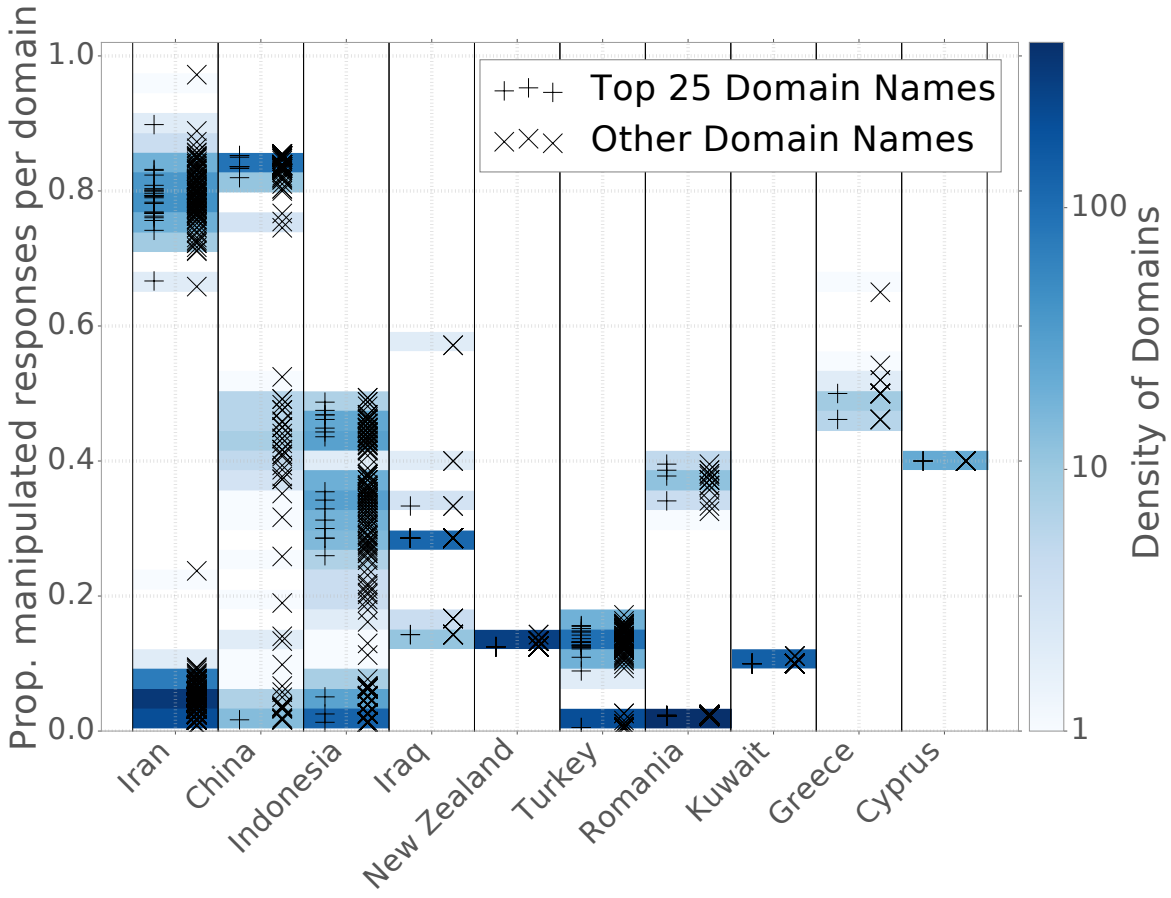


Figure 4.7: The fraction of resolvers within a country that manipulate each domain.

Similarly, one set of domains in China experiences manipulation by approximately 80% of resolvers, and another set experiences manipulation only half the time. In contrast, manipulation in Greece and Kuwait is more homogeneous across resolvers.

Heterogeneity across a country may suggest a situation where different ISPs implement filtering with different block lists; it might also indicate variability across geographic region within a country. The fact that manipulation rates vary even among resolvers in a certain group within a country may indicate either probabilistic manipulation, or the injection of manipulated responses (a phenomenon that has been documented before [9]). Other more benign explanations exist, such as corporate firewalls (which are common in the United States), or localized manipulation by resolver operators.

Ceilings on the percent of resolvers within a country performing manipulation, such as no domain in China experiencing manipulation across more than approximately 85% of resolvers, suggest IP geolocation errors are common.

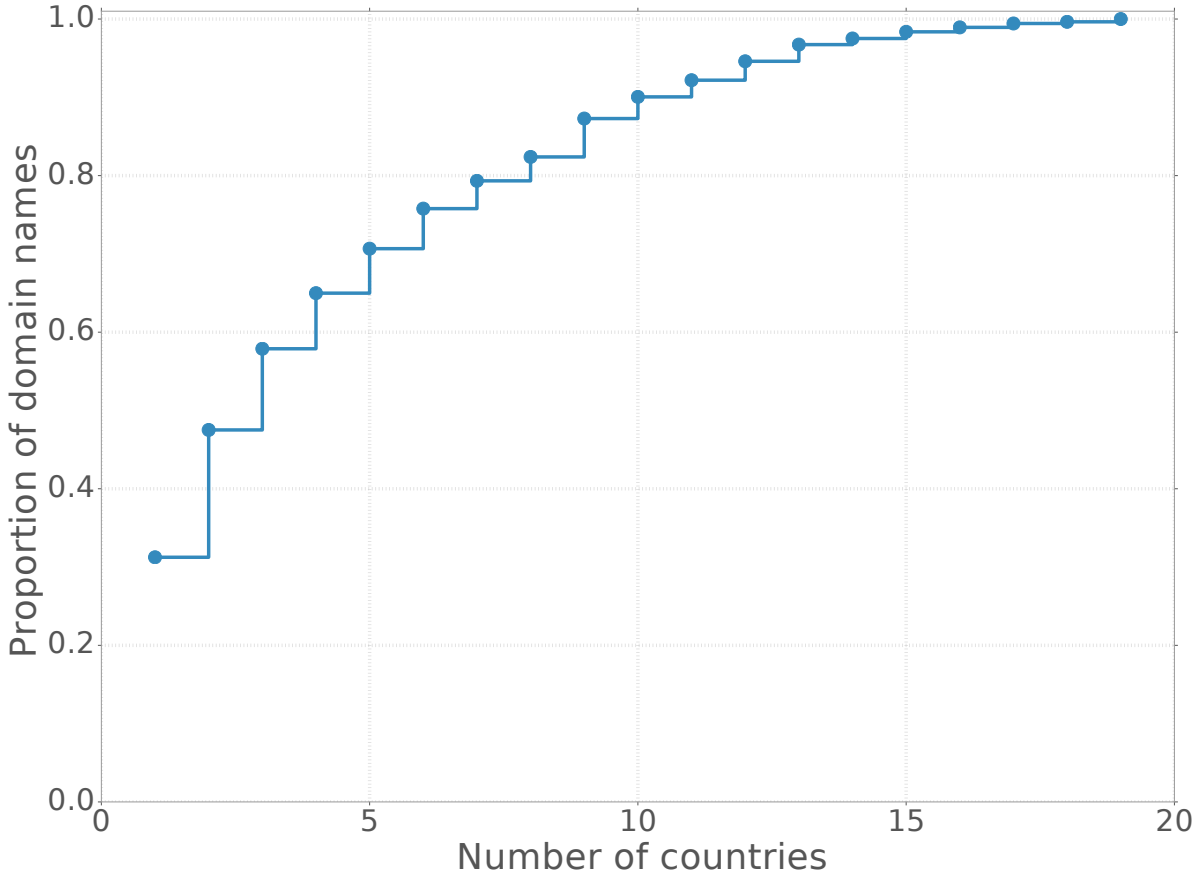


Figure 4.8: The number of countries (or dependent territories) that block each domain with observed manipulated responses, sorted by manipulation prevalence.

Commonly Manipulated Domains

Commonly manipulated domains across countries Many domains experienced manipulation across a range of countries. Figure 4.8 shows a CDF of the number of countries (or dependent territories) for which at least one resolver manipulated each domain. 30% of domains were manipulated in only a single country, while 70% were manipulated in 5 or fewer countries. No domain was manipulated in more than 19 countries.

Table 4.9 highlights domains that experience manipulation in many countries (or dependent territories). The 2 most manipulated domains are both gambling websites, each experiencing censorship across 19 different countries. DNS resolutions for pornographic websites are similarly manipulated, accounting for the next 3 most commonly affected domains. Peer-to-peer file sharing sites are also commonly targeted, particularly The Pirate Bay. The Tor Project [137] DNS domain is the most widely interfered with domain amongst anonymity and censorship tools, manipulated across 12 countries. Citizen Lab [30] also experienced manipulation across 4 countries. We note

Rank	Domain Name	Category	# Countries	# Res
1	www.pokerstars.com	Gambling	19	251
2	betway.com	Gambling	19	234
3	pornhub.com	Pornography	19	222
4	youporn.com	Pornography	19	192
5	xvideos.com	Pornography	19	174
6	thepiratebay.org	P2P sharing	18	236
7	thepiratebay.se	P2P sharing	18	217
8	xhamster.com	Pornography	18	200
9	www.partypoker.com	Gambling	17	226
10	beeg.com	Pornography	17	183
80	torproject.org	Anon. & cen.	12	159
181	twitter.com	Twitter	9	160
250	www.youtube.com	Google	8	165
495	www.citizenlab.org	Freedom expr.	4	148
606	www.google.com	Google	3	56
1086	google.com	Google	1	5

Table 4.9: Domain names manipulated in the most countries (or dependent territories), ordered by number of countries with manipulated responses.

that `www.google.com` is impacted across more countries than `google.com`, unsurprising since all HTTP and HTTPS queries to `google.com` immediately redirect to `www.google.com`; for example, China manipulates `www.google.com` queries but disregards those for `google.com`. This result underscores the need for domain datasets that contain complete domains and subdomains, rather than simply second-level domains.

We also note that commonly measured sites such as The Tor Project, Google, and Twitter, experience manipulation across significantly fewer countries than some sites. Such disparity points to the need for a diverse domain dataset.

China focuses its DNS manipulation not just on adult content but also major English news outlets, such as `nytimes.com`, `online.wsj.com`, and `www.reuters.com`. China is the only country observed to manipulate the DNS responses for these domains; it also censored the Chinese language Wikipedia domain.

Commonly manipulated categories Table 4.10 shows the prevalence of manipulation by CLBL categories. We consider a category as manipulated within a country if any resolver within that country manipulates a domain of that category. Domains in the Alexa Top 10K experienced the most manipulation; these domains did not appear in the CLBL, which highlights the importance of measuring both curated lists from domain experts as well as broad samples of popular websites. Although no single domain experiences manipulation in more than 19 countries, several categories

Rank	Domain Category	# Countries	# Resolv.
1	Alexa Top 10k	36	442
2	Freedom of expr.	35	384
3	P2P file sharing	34	394
4	Human rights	31	288
5	Gambling	29	377
6	Pornography	29	342
7	Alcohol and drugs	28	274
8	Anon. & censor.	24	303
9	Hate speech	22	158
10	Multimedia sharing	21	293
20	Google	16	234
34	Facebook	10	175
38	Twitter	9	160

Table 4.10: Top 10 domain categories, ordered by number of countries (or dependent territories) with manipulated answers.

experience manipulation in more than 30 countries, indicating that while broad categories appear to be commonly targeted, the specific domains may vary country to country.

To study how manipulated categories vary across countries, we analyzed the fraction of resolvers within each country that manipulate a particular category. The top categories vary extensively across countries. Table 4.11 shows the most frequently manipulated categories for the top 10 countries by normalized amounts of manipulation. The top category of manipulated content in Iran, "provocative attire," is not a category across any of the other top 10 countries. Manipulation of domains randomly selected from Alexa but not in the CLBL ("Alexa Top 10k") is prevalent across numerous countries, again reinforcing the need for diverse domain datasets. Anonymity and censorship tools are manipulated extensively across 85% of resolvers in China, but not across the rest of the top 10. Pornography and gambling sites are manipulated throughout.

4.5 Iris Summary

Internet censorship is widespread, dynamic, and continually evolving; understanding the nature of censorship thus requires techniques to perform continuous, large-scale measurement. Unfortunately, the state-of-the-art techniques for measuring manipulation—a common censorship technique—rely on human volunteers, limiting the scale and frequency of measurements. This work introduces a method for measuring DNS manipulation on a global scale by using as vantage points open DNS resolvers that form part of the Internet's infrastructure.

The major contributions of this chapter are: (1) Iris: a scalable, ethical system for measuring DNS manipulation; (2) an analysis technique for disambiguating natural variation in DNS

Country	Domain Category	% of Resolv.
IR	Provocative attire	90.98%
	Alexa Top 10k	90.16%
	Freedom of expr.	90.16%
CN	Alexa Top 10k	85.48%
	Freedom of expr.	85.48%
	Anon. & censor.	85.48%
ID	Pornography	57.50%
	Alexa Top 10k	56.25%
	P2P file sharing	52.50%
IQ	Political Blog	57.14%
	Alexa Top 10k	28.57%
	Freedom of expr.	28.57%
NZ	Alexa Top 10k	12.50%
	Freedom of expr.	12.50%
	P2P file sharing	12.50%
TR	Alexa Top 10k	18.23%
	Freedom of expr.	17.71%
	Pornography	16.67%
RO	Alexa Top 10k	37.78%
	Gambling	37.78%
	Freedom of expr.	2.22%
KW	Alexa Top 10k	10.00%
	Freedom of expr.	10.00%
	P2P file sharing	10.00%
GR	Gambling	50.00%
	Alexa Top 10k	46.15%
CY	Alexa Top 10k	40.00%
	Gambling	40.00%

Table 4.11: Breakdown of the top 3 domain categories experiencing manipulation, per country. Countries are ordered by the relative amount of manipulated responses for that country. Both Greece (GR) and Cyprus (CY) only experience manipulated responses across 2 categories.

responses (e.g., due to CDNs) from more nefarious types of manipulation; and (3) a large-scale measurement study that highlights the heterogeneity of DNS manipulation, across countries, resolvers, and domains. Notably, we find that manipulation is heterogeneous across DNS resolvers even within a single country. Iris supports regular, continuous measurement, which will ultimately facilitate tracking DNS manipulation trends as they evolve over time; our next step is to operationalize such measurements to facilitate longitudinal analysis.

Chapter 5

Censorship Discussion and Conclusion

In Part I we developed *Augur* (Chapter 3) [116, 117] and Iris (Chapter 4) [119, 120], two methods and accompying systems that allow us to ethnically, and without the need for volunteers, measure censorship behavior globally at the TCP/IP and DNS layers. From our initial deployments, we verified their accuracy, identified new censored content, and described heterogenous censorship beheavior within countries, each at a global scale.

With the existance of tools like Augur and Iris, we can now begin to develop longitudinal, continuous views of Internet censorship, censors and their behavior. This can provide us with the ability to answer numerous questions previously difficult to address. These include:

- **How do Internet censorship trends vary over time?** Understanding how blocking changes over time allows us to identify events and actions that influence a censor's behavior. These trends are critical, both qualitatively and quantitatively, for a range of social science research aimed at understanding and addressing censorship.
- **How do Internet censors respond to circumvention, and on what time scale?** Related to censorship trends is how censors respond to active circumvention of their efforts. Understanding how censors engage and on what time scales provides empirical grounding for evaluating various defenses and enables constructing more effective circumventions.
- **Does censorship vary between regions within countries?** Examining how censorship varies between regions provides insights into how censorship systems are deployed, both technically and organizationally. These insights can better inform circumvention and policy interventions. Augur and Iris's use of multiple vantage points across a range of networks within a country allows us actively explore and understand these deployments beyond our initial results.

Being able to provide a comprehensive empirical footing for these questions can enable new lines of research as well as serve as an invaluable resource for social scientists.

Next steps

The continuous, widespread measurements that we can collect with these techniques can complement anecdotes, news reports, and policy briefings to ensure that we can support future assessments of Internet filtering with sound, comprehensive data. Part of this transition to practice involves further developing the system that we have designed to facilitate ongoing operation, including automating the validation of the measurements that we collect and the correlation with other datasets and tools. We aim to ultimately compare results produced by multiple methods, including datasets from volunteer-based measurement platforms.

Part II

Understanding Advertising Abuse

Chapter 6

Advertising Abuse Introduction and Related Work

6.1 Introduction

Profit drives modern cybercrime. Investments in malware, botnets, bullet-proof hosting, domain names, and other infrastructure must all be justified by the greater revenue brought in by the scams that use them. Thus, scammers relentlessly innovate to identify more lucrative niches to maximize their returns (e.g., counterfeit goods, fake anti-virus, ransomware, credit card theft, cryptocurrency mining, etc.). Monetization, however, also presents some of the greatest risks, since it represents both a single point of vulnerability in the business model and a potential means of forensic attribution [69, 89, 99]. Thus, the ideal monetization strategy for an Internet scammer would not only tap into great wealth, but would also effectively launder the money trail as well. As we will describe in this chapter, the advertising ecosystem is remarkably well-suited to this need, and advertising abuse provides perhaps the *non plus ultra* of all such monetization schemes.

First and foremost, online advertising represents an enormous revenue stream, generating over $88 billion in revenue in 2017 and growing at over 20% per year [122]. Moreover, online advertising is a low-friction market designed to engage the broadest possible set of participants, and thus presents few barriers-to-entry for potential bad actors. Unsurprisingly, criminal groups have developed a range of techniques for generating synthetic advertisement clicks for profit at the expense of legitimate advertisers and ad networks [38]. Indeed, such *click fraud* accounts for as much as 10% of all advertising clicks, potentially defrauding advertisers of hundreds of millions of dollars annually.

As a further difficulty, the ecosystem of online advertising is highly complex, and while occasionally traffic flows directly from publisher to advertiser, in the common case instead a vast array of middlemen—syndicators, subsyndicators, traffic aggregators and resellers—separates the two endpoints of each ad click. While the path of such a click on the Internet is nominally visible (a chain of redirects from one domain to the next), the chain of payment remains largely opaque. Those on the buy-side and the sell-side of the traffic ecosystem negotiate their contracts bilaterally, with a wide range of terms and conditions. Thus, no single party comes remotely close to having

comprehensive visibility of who gets paid what for any given click. Finally, as we describe, click fraud platforms can engage with a wide range of ad networks, who in turn *mix* this traffic with other, more legitimate sources, further complicating any forensic analysis.

To better understand and defend against advertising abuse broadly, in this chapter we systematically examine multiple monetization strategies, attacks, and advertising abuse botnets—from both an external and internal ad network perspective.

We begin in Chapter 7 by examining the operation and underlying economic models of two families of common clickbots, "Fiesta" and "7cy." By operating the malware specimens in a controlled environment we reverse-engineer the protocols used to direct the clickbots in their activities. We then devise a *milker* program that mimics clickbots requesting instructions, enabling us to extract over 360,000 click fraud directives from the clickbots' control servers. We report on the functioning of the clickbots, the steps they employ to evade detection, variations in how their controllers operate them depending on their geographic locality, and the differing economic models underlying their activity.

Next, in Chapter 8 we explore the timeline and history of ZeroAccess (ZA). ZeroAccess was one of the largest botnets ever in operation, controlling an estimated 1.9 million infected computers at its peak [109]. ZeroAccess was particularly known for monetization primarily via click fraud, with losses to advertisers estimated at $2.7 million per month [149].

Chapter 9 explores the technical functionality of the ZeroAccess malware itself. Using a combination of code analysis and empirical measurement, we document the distinct command-and-control protocols used by each module, their infrastructure, and how they operate to defraud online advertisers. This worked served as the legal basis of a law enforcement-driven takedown (Section 8) of the botnet [90, 104].

Chapter 10 further expands our understanding of ZeroAccess-driven advertising abuse by leveraging an insider perspective. This chapter illuminates the intricate nature of ZeroAccess fraud using a broad range of data sources, including peer-to-peer measurements, command-and-control telemetry, and contemporaneous click data from one of the top ad networks. From this multifaceted approach we construct a view into the scale and complexity of click fraud operations. By leveraging the dynamics associated with the takedown of ZeroAccess in December 2013, we employ this coordinated view to identify "ad units" whose traffic (and hence revenue) primarily derived from ZeroAccess. While it proves highly challenging to extrapolate from our direct observations to a truly global view, by anchoring our analysis in the data for these ad units we estimate that the botnet's fraudulent activities plausibly induced advertising losses on the order of $100,000 per day.

Another facet of the ecosystem of advertising abuse is *ad injection*, an attack where ads are inserted into a user's browsing by software running on their system [135]. This attack harms tens of millions of users and leverages the complexity of multi-hop ad reseller chains to *launder* fraudulent traffic, masking it from the visibility of ad networks. In Chapter 11 we work with Google, exploring the population of intermediaries and advertisers that support the abuse ecosystem through analysis of injected click (revenue) chains, thus identifying structural "choke points" that were leveraged for defense.

6.2 Related Work and Background

As background, we provide an overview of the advertising ecosystem on the Web, how attackers defraud Web advertising networks, specifically the methods by which ZeroAccess and other botnets have perpetrated click fraud at scale, and the overall ecosystem of ad abuse malware and botnets.

Web Advertising

Advertising on the Web works in terms of arrangements between *advertisers*, who wish to display promotional content, and *publishers*, who receive visits from users who could potentially view and respond to that content. Publishers receive payment for displaying the advertiser's content, which can consist of text, images, video, or other interactive (e.g., Flash-based or JavaScript-based) media. Advertisements generally include links to the advertiser's site to allow interested users to directly engage with the advertiser by clicking on the advertisement and visiting the site.

In practice, advertisers and publishers often do not deal with each other directly. Instead, each contracts with an *advertising network* that coordinates ad placement between many advertisers and publishers. In a traditional arrangement, an advertiser buys a given volume of advertising from the ad network, usually also specifying a set of *keywords* defining the context in which to show the ad. Publishers then join an advertising network and display ads provided by the network.

When a user issues a search query, the resulting page includes organic search results, for which the linked Web sites do not pay the search engine, as well as paid search ads, for which the linked Web sites (the advertisers) pay the search engine for inclusion. These ads are typically placed above the organic results or alongside on the right. They are formatted similarly to search results except for a darker shade background and the word "Ad" or "Sponsored" somewhere nearby.

Search engines select ads based on the user's search query. The search query is distilled into a group of keywords after normalizing to remove misspellings and resolve ambiguities.

Search ad syndication networks. Search engines partner with thousands of Web sites, services, and applications—collectively called publishers—to extend the reach of the search engine's advertisers to users beyond the search engine. Publishers include blog sites, news sites, niche search engines, ad-supported browser addons, and other ad networks.

The publisher sends the user's search query from the publisher's Web site or app, or in the case of blogs and news sites, the context of the article the user is reading, to the search engine's ad server to retrieve relevant ads in exchange for a cut of the advertising revenue. The publisher can display relevant ads by embedding JavaScript provided by the ad server, which directs the user's browser to fetch the ads as illustrated in steps 1–6 in Figure 6.1. Alternatively, the publisher can directly fetch relevant ads through server-to-server communication between the publisher and the ad server (not shown in the figure). If the user clicks the ad, the user is taken to the advertiser's site through a series of redirects as shown in steps 6–11 in the figure. Each interaction at the ad server is logged along with the publisher identifier for the publisher responsible for the traffic.

Publishers can, in turn, (sub)syndicate to other downstream publishers according to the search ad network's policy. The downstream publisher requests ads from the intermediate publisher,

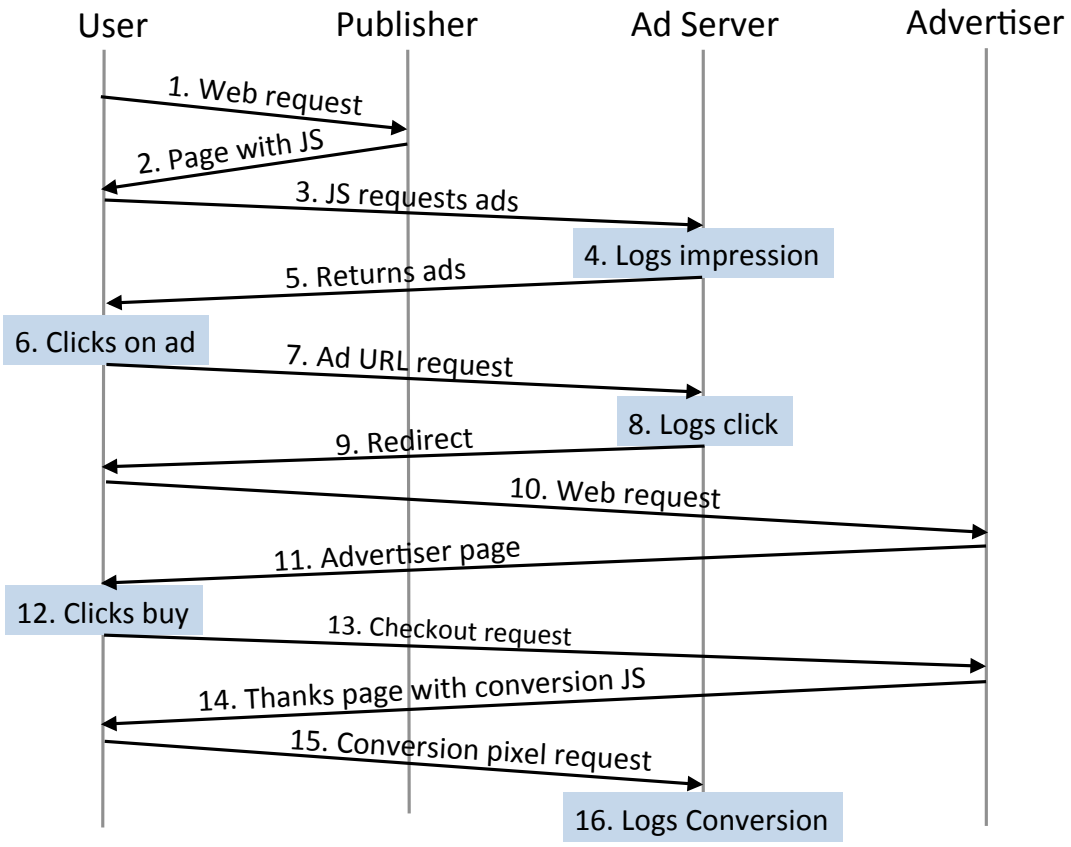


Figure 6.1: Anatomy of a typical ad click, showing the various HTTP requests associated with a user clicking on an advertisement, leading them to an advertiser's *landing page*, and from there possibly to additional interactions.

which then fetches them from the ad server. The search engine pays the intermediate publisher a cut of the ad revenue. The intermediate publisher pays the downstream publisher a smaller cut and retains the rest in exchange for the service it provided. Thus the search engine does not directly deal with the downstream publisher, and in many cases never learns of its existence. Subsyndication arrangements exist to help search ad networks scale to hundreds of thousands of publishers by distributing the management overhead down tiers of aggregators.

For example, Google and Bing syndicate search ads to other search engines including Infospace and Yahoo, respectively [60, 103]. Infospace and Yahoo show these ads on Web sites they own and operate, but also subsyndicate to smaller networks like Publishers Clearing House and Chitika respectively, which then further subsyndicate to yet smaller publishers.

Revenue share. Search ads are typically charged on a cost-per-click (CPC) model, i.e., the

advertiser pays only if the ad is clicked. For clicks on ads shown by a syndicate publisher, the search engine typically retains 30% of the amount charged to the advertiser, and pays a 70% cut to the publisher [83]. Publishers that sub-syndicate set their own revenue sharing agreements with their downstream publishers.

Click Fraud

Click fraud is the practice of fraudulently generating clicks on ads without any intention of fruitfully interacting with the advertiser's site. As a result, advertisers lose money, receiving no return on their investment.

There are two primary motivations for click fraud. First, a malicious advertiser can employ click fraud targeting a competitor's ad to deplete their advertising budget [141]. A stronger motivation, however, lies with publishers, who directly profit from ads clicked on their site. Ad networks also profit from click fraud, though reputable ad networks that want to maintain long-term relationships with advertisers will presumably attempt to identify and weed out click fraud activity. Note that it is hard for an ad network to prove that a particular click was fraudulent because it is equivalent to guessing the intent of the user behind the click.

Revenue sharing with (sub)syndicate publishers creates an economic incentive for these publishers to fraudulently attract clicks on ads shown by them. While a well-known publisher would lose reputation if caught engaging in click fraud, followed by financial losses if ejected from the syndication network, this disincentive does not exist for less well-known publishers. Less reputed publishers may not have a reputation to protect, or may be able to reattach to the syndication network at a different point if ejected. As a result, click fraud from relatively unknown publishers has historically been rampant [132].

Click Fraud techniques have evolved considerably in the past several years. The direct approach of hiring people to click on ads (termed click farms) [58], or running stand-alone scripts that repeatedly retrieve the URLs associated with ads to simulating user clicks (stand-alone bots) [96], are now easily detected by an ad network's defenses. More complex approaches include search engine hijacking [125], where a malicious in-browser plugin replaces ad links in results returned for user searches by other ads, and the rise of click fraud botnets like ZeroAccess that coordinate large numbers of malware-infected hosts to fetch and click ads unbeknownst to the user.

A number of case studies have chronicled the evolution of click fraud botnets over the years, ranging from the early Clickbot.A botnet [37] to the TDL-4 botnet [125], the Fiesta and 7cy botnets [105], and ZeroAccess itself [118, 149]. Security researchers have similarly documented more attacks in blog posts and in whitepapers [45, 51, 72]. Unlike previous studies, however, we analyze the ZeroAccess botnet primarily from the perspective of the advertising network, supplemented with insight from operational data derived from its P2P infrastructure and use of DNS. As a result, we are able to present the first comprehensive characterization of monetization, distribution, and activity of a massive click fraud botnet.

At the same time, malicious (sub)syndicated publishers have become better at avoiding detection or identification. Using techniques such as referrer cloaking [42], or fetching ads through other publishers that use the direct server-to-server mechanism, many sub-syndication publishers

remain completely anonymous. Intermediate publishers complicit in this activity blend in traffic from non-malicious small publishers to present a cleaner appearance in the aggregate to their syndication parent. This blending results in even reputed publishers unwittingly facilitating click fraud.

Click fraud mitigation and smart-pricing. Due to the large number and relative anonymity of publishers in the (sub)syndicate network, search engines rely primarily on automated means, supplemented with limited manual investigations, to protect their advertisers from click fraud. Rule-based techniques [84, 141], correlation analysis [101, 102, 152], and bluff ads [66] have focused on detecting clicks from infected users before the advertiser is charged. Clustering [38] and anomaly detection [39] have focused on detecting malicious publishers. Not-A-Bot (NAB) [65] combats bot activity through detection mechanisms at the client. In the NAB system, the client machine has a trusted component that monitors keyboard and mouse input to attest to the legitimacy of individual requests to remote parties. In addition to NAB, Juels et al. likewise propose dealing with click fraud by certifying some clicks as "premium" or "legitimate" using an attester instead of attempting to filter fraudulent clicks [77]. Smart-pricing [59] takes an economic approach to mitigating the impact of click fraud. At a high level, smart-pricing maintains a normalization factor between 0 and 1 for each publisher based, in part, on the probability of *conversions* generated by traffic sent by that publisher [61]. Conversions are actionable business results as defined by the advertiser, such as an online sale or newsletter sign-up. Advertisers inform the ad server of a conversion by embedding JavaScript provided by the ad server on the Web page corresponding to the conversion (Figure 6.1, steps 12–16). The ad server uses cookies to link the conversion to an earlier ad click, which is then associated with the publisher that was responsible for the click. For a simplistic illustration, consider otherwise identical publishers X and Y that both send 100 legitimate users, but X additionally blends in 100 fraudulent clicks. Consider further that 10 legitimate users convert in each case. X therefore has half the conversion probability of Y (5% vs. 10%). Smart-pricing normalizes CPC for X to effectively be half of that for Y, such that the advertiser has the same effective cost per conversion.[1] Implicit in the smart-pricing mechanism is the assumption that click fraud traffic will not result in conversions, an assumption that the Serpent module (Section 8) of ZeroAccess specifically tries to defeat.

Our work informs this debate with the latest generation of click fraud botnets, and identifies potentially fruitful directions for building the next generation of mitigation capabilities.

Malware and botnets

Clickbots. The only prior academic work analyzing the functionality of a botnet performing click fraud focused on a bot named Clickbot.A [37]. Clickbot.A conducted low-noise click fraud through the use of syndicated search engines. Daswani et al. employed a combination of execution, reverse-engineering, and server source code analysis to determine how Clickbot.A performed fraud. The clickbot used compromised web servers for HTTP-based C&C, and restricted the number of clicks

[1] In practice, smart-pricing takes multiple features into account, and applying the normalization factor given dynamic bidding and publisher diversity is not as straight-forward

performed for each bot, presumably to limit exposure to the ad agency. In addition to describing the botnet behavior, the investigators estimate the total fraud against Google using economic aspects of syndicate search and click pricing. Our work analyzes multiple clickbot specimens to understand the changes in both the economic model and bot operation in two common clickbots. We expect that criminals are constantly improving bot technology in order to remain competitive against ad agencies that improve their fraud detection. Throughout this work we use Clickbot.A as a reference for comparison.

Detecting Automated Search. Researchers have dedicated considerable effort to methods for differentiating search queries from automated and human sources. Yu et al. observe details of bot behavior in aggregate, using the characteristics of the queries to identify bots [152]. Buehrer et al. focus on bot-generated traffic and click-through designed to influence page-rank [19]. Kang et al. propose a learning approach to identify automated searches [80]. These efforts do not examine bot binaries or C&C structure, focusing instead on techniques for the search engine to identify automated traffic.

Botnets. There is extensive work examining botnets and reverse-engineering bots and their C&C protocols [1, 21, 25, 26, 67, 74, 121]. Dispatcher is a technique that automatically reverse-engineers botnet C&C messages and was used to uncover details in the MegaD spamming botnet C&C protocol [21]. In a later work, Cho et al. used Dispatcher to conduct an extensive infiltration of the MegaD botnet, developing milkers to determine the C&C structure and mine data about the botnet's internal operations [26]. We employ a similar milking technique to explore the C&C structure of the clickbots under study.

Chapter 7

What's Clicking What? Clickbot Techniques and Innovations

7.1 Introduction

Online advertising forms a vital part of the modern Internet economy. Millions of websites profit from an ecosystem of advertising networks and syndication chains. This widespread adoption of internet advertising has given rise to systematic fraud. The percentage of fraudulent ad clicks, called *click fraud*, has steadily increased over recent years. Estimates suggest the fraud-rate is as high as 13.5% [54].

In the predominant form of click fraud, a malicious party sets up a website filled with ads and deceives an advertising network into registering ad clicks, earning revenue for each click.[1] *Clickbots*, malware that automatically clicks on ads, can produce this fraudulent traffic. A challenge for clickbot operators is producing traffic in such a way that advertising agencies do not detect it as non-human or fraudulent.

In this chapter, we present an analysis of clickbot techniques and the associated infrastructure that supports click fraud. We obtained samples of two clickbot families, which we named "Fiesta" and "7cy," in order to study the operation of clickbots. We executed the binaries in a controlled environment to prevent harmful side effects, such as actually participating in click fraud. By monitoring the controlled execution of the bots, we reverse-engineered their command and control (C&C) protocols to determine how the bots respond to the commands they receive. This analysis enabled us to develop C&C servers and websites for the bots to interact with, allowing greater exploration of bot behaviors. We then devised a *milker* program that mimics a clickbot requesting instructions, enabling us to extract 366,945 click fraud directives from the clickbots' control servers.

Throughout our analysis, we compare both families of clickbots to Clickbot.A [37] in order to illuminate how clickbots have evolved over time. At the time of publication, we found two

[1] In a second form of click fraud, a malicious party deliberately focuses clicks on a competitors advertisements in an attempt to exhaust that party's advertising budget [92].

major innovations. The first regards the underlying economic model used by the Fiesta family. In this model a middleman has emerged, acting as a layer of abstraction between ad syndicates and the clickbots that generate traffic. This middleman provides an intermediary between the traffic originator (bots and users) and the ad syndicates. Fiesta clickbots generate traffic that flows towards this middleman and is then laundered through a series of ad syndicates in an effort to hinder fraud detection.

The second innovation concerns complex behavior that attempts to emulate how humans browse the web. 7cy mimics a human browsing the web by both randomizing the click targets as well as introducing human-scale jitter to the time between clicks. Through the use of our milker we also discover that 7cy control servers distribute fraud directives that vary by the geographic region of the bot. Thus, while Fiesta generally uses indirection to accomplish click fraud, 7cy uses random clicks, timing, and location-specific behavior to ostensibly present more realistic browsing behavior.

In Section 7.2 describes our methodology for executing bots in a safe environment. Section 7.3 discusses the Fiesta clickbot in depth, and Section 7.4 looks at 7cy. We discuss potential defenses and then summarize in Section 7.5.

This chapter is based on work that appeared in Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA) [105].

7.2 Methodology

In this section we outline our environment and procedures for executing clickbots without risk of malicious side effects. We studied two "families" of clickbots, Fiesta and 7cy, within our experimental framework. [2] Since we obtained samples that did not have useful or consistent anti-virus labels we took the names Fiesta and 7cy from domain names the bots visited while performing click fraud.

We obtained the Fiesta and 7cy samples by infiltrating several malware Pay-Per-Install (PPI) services as part of a larger study on malware distribution [20]. PPI services use varied means to compromise machines and then distribute malware to the compromised hosts in exchange for payment on the underground market [20]. We used behavioral characteristics to group multiple harvested binaries representing different versions of a given family of malware. We selected Fiesta and 7cy for further analysis because their connection behavior and content was the most interesting. A third potential clickbot family remains unanalyzed.

We executed the clickbots in virtual machines hosted on VMware ESX servers. A central gateway, implemented using Click [85], moderates network access for the VMs. The gateway routes each outbound connection to a "containment server" that decides on a per-flow basis whether traffic is forwarded, dropped, rewritten, or reflected back into the contained network. The containment

[2] The MD5 hashes of the Fiesta specimens are c9ad0880ad1db1eead7b9b08923471d6 and 5bae55ed0eb72a01d0f3a31901ff3b24. The hashes of the 7cy specimens are 7a1846f88c3fba1a2b2a8794f2fac047, b25d0683a10a5fb684398ef09ad5553d, 36ca7b37bb6423acc446d0bf07224696, and 782538deca0acd550aac8bc97ee28a79.

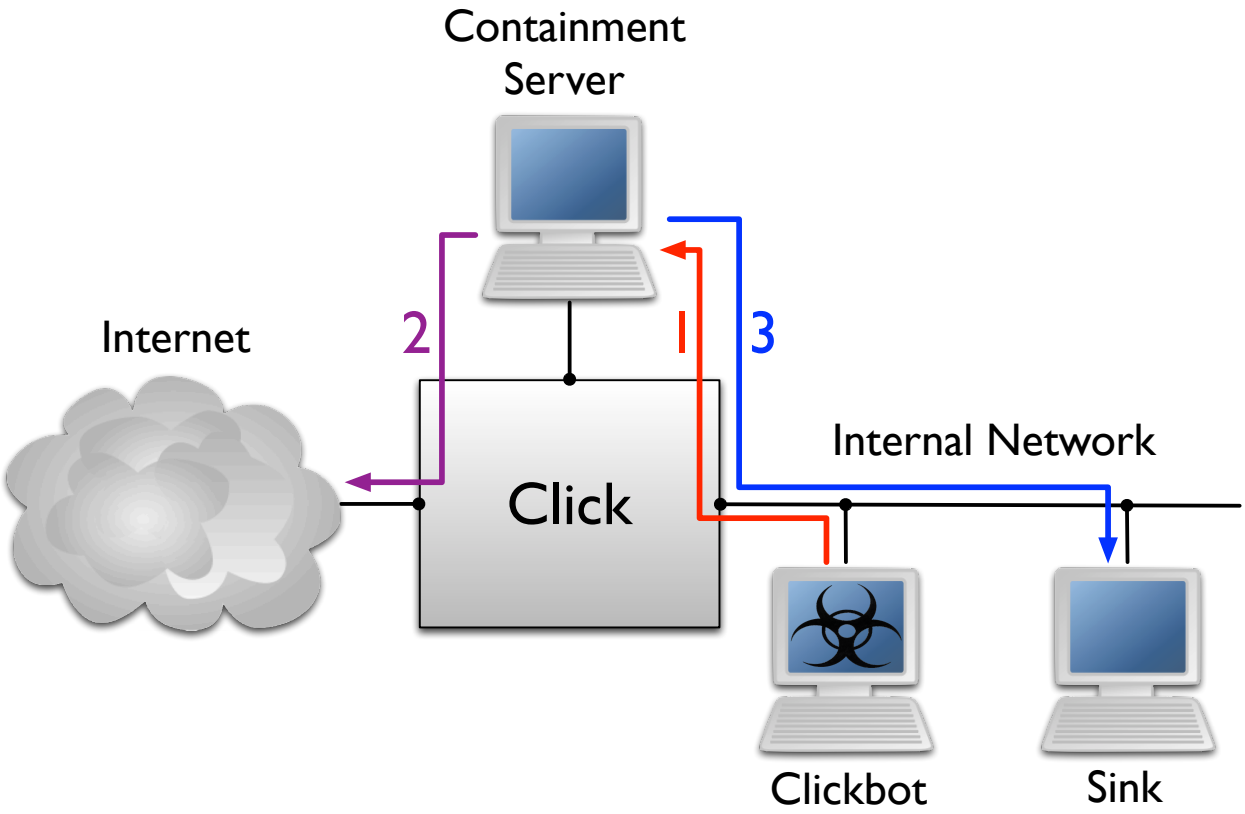


Figure 7.1: Our basic containment architecture, showing how a containment server can interact with an infected VM and a "sink" server. The clickbot's communication is routed through the containment server (1), which can allow the traffic (perhaps modified) out to the Internet (2), or redirect it back into the farm to the sink (3).

server makes these decisions based on packet header information as well as packet content. Figure 7.1 shows a simplified view of this approach.

Given this architecture, we implemented containment policies that allowed us to execute the clickbot specimens safely. These policies required understanding the basic behavioral patterns of the bots and the other parties involved. This was an iterative process in which we repeatedly examined the bot behavior connection by connection, starting from a default-deny policy. Each step in this iterative process involved manually examining connections and packet payloads by hand to verify the nature of the communication. In some cases, this meant examining data logs, and in other cases it involved manually visiting websites. We white-listed connections deemed safe, and then restarted the bot in order to identify the next communication.

We needed the capability to replay pre-recorded communication and interact with the bot entirely within the farm in order to explore each clickbot's behavior and C&C protocol. Therefore, we built a special HTTP "sink" server that impersonated the destinations of outbound clickbot flows. This server allowed us to respond to network communication using a server under our con-

trol rather than releasing the traffic from the farm or dropping the flow. The sink server accepted all incoming connections and returned pre-defined responses as a function of the HTTP header data. Since the bot used HTTP for C&C as well as web browsing, we used the same HTTP server to simulate both C&C and web traffic. Initially, we replayed previously seen C&C responses. Then, we manually explored and perturbed the plain-text C&C protocol and fed these modified responses back into the bots within the farm. Using this technique we reverse-engineered much of the protocol used for both bot families. As we understood more of the C&C protocols, we modified the responses to change the behavior of the bot. Using our capability to replay previously observed communications and explore new communication variants, we accelerated our analysis and developed a broader understanding of the clickbots' behavior.

7.3 The Fiesta Clickbot

We selected the Fiesta clickbot as the first specimen for in-depth analysis. The primary innovation we discovered during this evaluation is the monetization opportunity created by separating traffic generation (bots) from ad network revenue. In this model intermediate pay-per-click (PPC) services "launder" clicks generated by bots and then deliver them to ad networks. The intermediate PPC service abstracts the botmaster from advertising networks and is a new development since the investigation into Clickbot.A. We have not found any record of the security community studying the Fiesta clickbot. [3] In this section we describe Fiesta's behavior and structure, then discuss an economic model for click fraud based on our observations. We conclude with a discussion of the bot's prevalence.

C&C Structure

There are two key players in the operation of Fiesta: a botmaster running a C&C server, and the self-described "Fiesta Pay-Per-Click (PPC) Profitable Traffic Solution." Fiesta PPC operates a store-front complete with signup and forum services. Although we named this clickbot Fiesta after the Fiesta PPC service, we believe the PPC service and the botmaster to be separate entities with different economic incentives. This relationship is discussed further in Section 7.3.

Immediately upon infection the Fiesta bot establishes an HTTP connection with the bot's C&C server. The server's IP address is statically coded into the binary and remains constant for the lifetime of the bot. Using this server, the bot performs a one-time connection that informs the C&C server that a new infection has occurred. After this initial connection the bot settles into a constant cycle of click fraud. Figure 7.2 gives a high-level overview of Fiesta's behavior for a single click. One click constitutes one act of click fraud and involves multiple queries to the C&C server and multiple interactions with web pages. We observed our Fiesta bot performing about three such clicks per minute.

[3] One variant observed was classified by an Anti-Virus vendor as the parasitic Murofet trojan [15]. We believe this to be a case of mistaken identity resulting from a standard Fiesta binary becoming infected and playing host to an instance of Murofet.

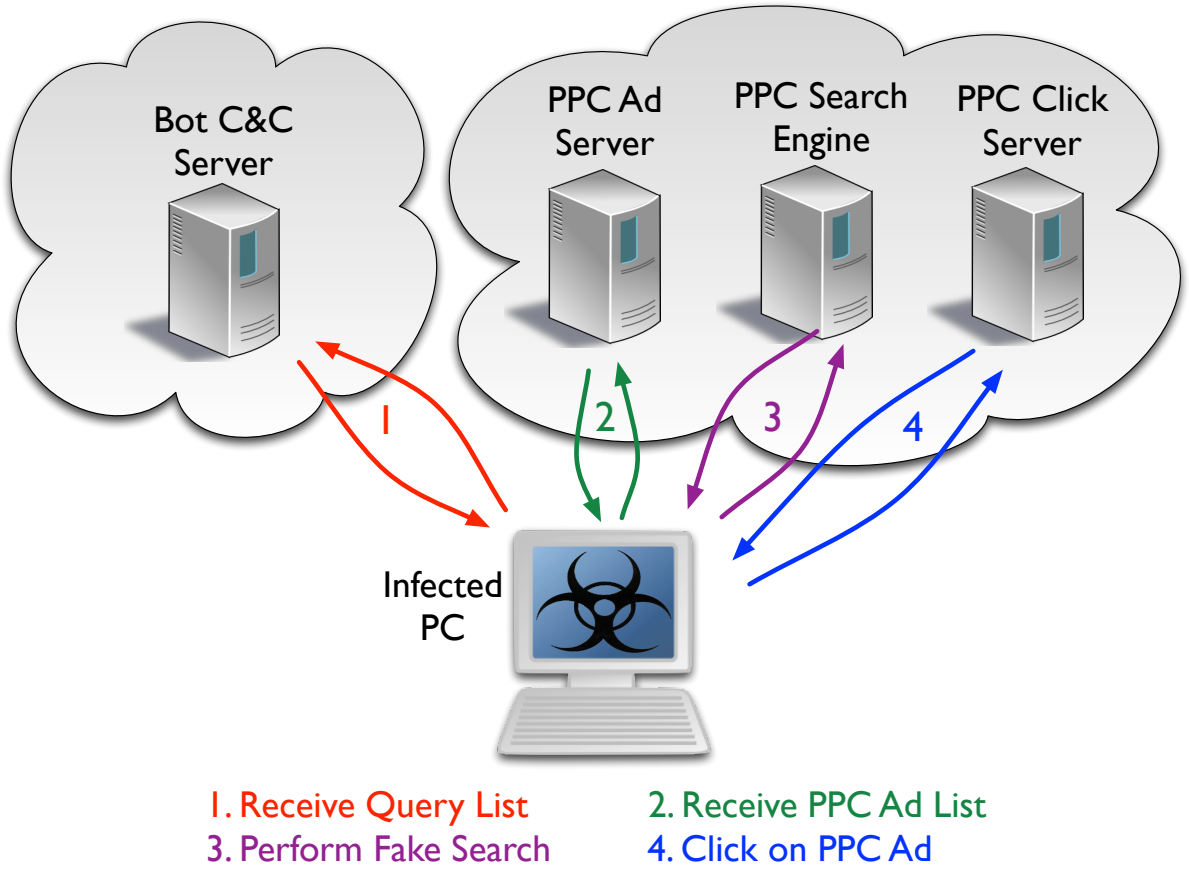


Figure 7.2: The basic behavioral architecture of the Fiesta clickbot. Communication occurs in the order specified by the numeric labels. This pattern repeats indefinitely.

nerdy shirts	cruise special	fifa world cup qualifiers
potato canon	among the hidden solution	kitchen aid dishwashers
ftv plus	scooby doo online games	yahoo real estate
online video	card cheap credit machine	oakland newspaper
cheap insurance life uk	camera disposable pentax	tapes on self help
celtic names	debt and consolidation	bozeman schools. mt
justin om	dallas nursing institute	anniversary gifts by year
vxb bearings	discount hotel booking	station nightclub fire video

Table 7.1: A sample of search query terms returned to the bot by the Bot C&C Server during a single exchange.

```
<?xml version="1.0" encoding="UTF-8"?>
<records>
 <query>u2 tour</query>
 ...
 <record>
  <title>Looking for u2 tour?</title>
  <description>Find u2 tour here!</description>
  <url>http://u2-tour.com</url>
  <clickurl>http://CLICK_SERVER/click.php?c=UNIQUE_ID</clickurl>
  <bid>0.0004</bid>
  <fi>52</fi>
 </record>
   ...
 <record>
   <title>Style Fashion Show Review</title>
   <description>Chanel</description>
   <url>http://www.style.com</url>
   <clickurl>http://CLICK_SERVER/click.php?c=UNIQUE_ID</clickurl>
   <bid>0.0023</bid>
   <fi>39</fi>
 </record>
 ...
</records>
```

Figure 7.3: Sample Fiesta ad C&C returned in response for a fake search for "u2 tour." The C&C syntax is abbreviated as necessary for space.

A fraudulent click begins with Fiesta requesting a list of search query terms from its C&C server, shown as step 1 in Figure 7.2. In response the bot receives several hundred varied terms; Table 7.1 shows some samples. We observed these terms changing frequently, appearing arbitrary in nature, and containing typographical errors. Once the bot receives this query list, it randomly selects one term that it will use for the remainder of this click.

After the bot selects a search query, the bot begins communicating with the Fiesta PPC service, labeled step 2 in Figure 7.2. Initially the bot performs a request to receive ads that correspond to the selected search query. In response, the PPC Ad Server returns approximately 25 ads in XML format. Figure 7.3 shows an example of the PPC Ad Server XML response. Information contained in each record includes an ad URL, title, keywords, and "bid." The PPC Ad Server returns ads that vary greatly. Some ads directly relate to the search, while others appear random. The bot selects one of the ads at random from the PPC Ad Server response, biasing its selection towards high bid values. After selecting an ad, the bot performs a search for the original search query at a search engine operated by the PPC service. The bot then informs the search engine which ad it is about to click via a separate HTTP request (step 3 in Figure 7.2). Lastly, the bot will contact the PPC Click Server and actually perform the ad click (step 4 in Figure 7.2).

The PPC Ad Server returns ad URLs that point to the PPC Click Server. Each ad URL contains a unique 190-character identifier that is used when the bot issues an HTTP request to the PPC Click Server to signal a click. The PPC Click Server responds to all clicks with HTTP `302` redirect responses, beginning the fraudulent click. Figure 7.4 shows the process that occurs once the bot issues a request to the PPC Click Server, with arrows representing HTTP redirects. A single click

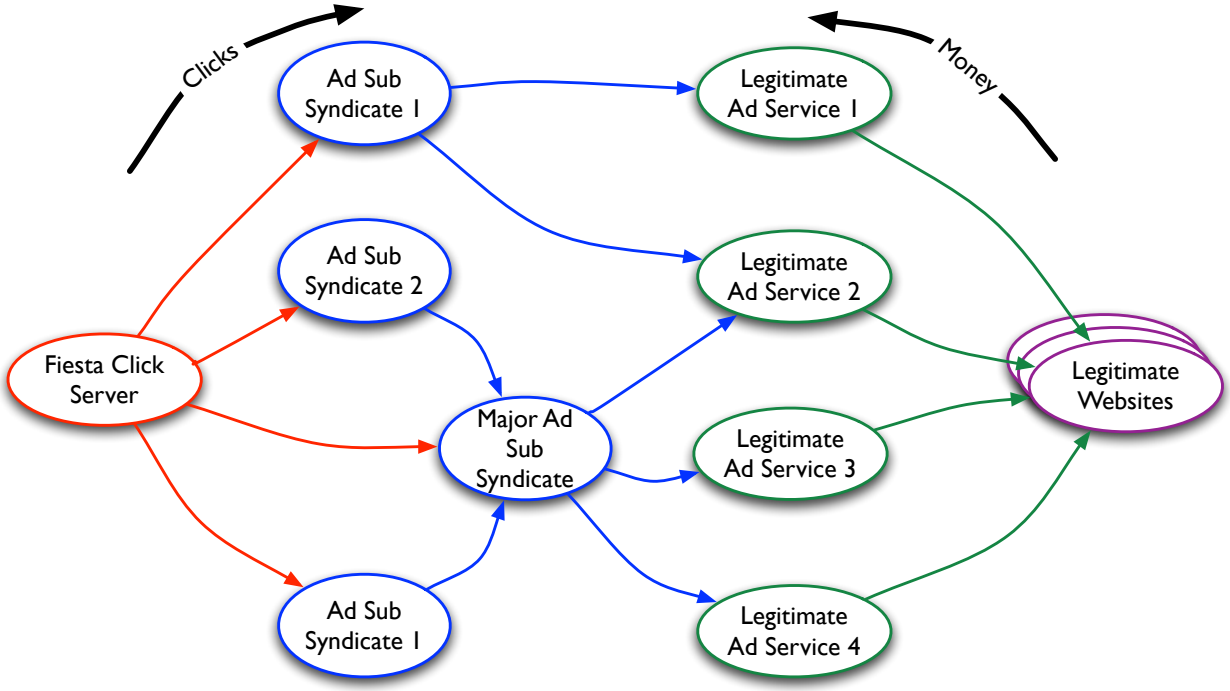


Figure 7.4: The expanded Fiesta click redirection chain. This graph represents the possible redirection paths beginning with a click on the Fiesta click-server and ending at a legitimate website. One click will take one path through the graph. Redirections flow from left to right, and money flows from right to left.

will cause the bot to receive redirects to three or four different locations, eventually settling on a legitimate website such as `style.com` or `accuweather.com`. The resolution of this redirection chain completes a single act of click fraud.

We believe the sites directly linked to the PPC Click Server along the redirection chains in Figure 7.4 are advertising sub-syndicates (i.e., entities that show ads generated by other ad networks in exchange for some portion of generated revenue) that have entered into syndication agreements with legitimate advertising services. The legitimate advertising services include *BidSystems* and *AdOn Network*. We believe some of the ad sub-syndicates are illegitimate based on other services hosted on the same IP addresses, as well as the frequency with which the sub-syndicate's IP addresses appear in malware reports.

Interestingly, the Fiesta bot issues requests to the Fiesta Ad Server with HTTP referrers from Fiesta search engines, yet performs searches *after* issuing ad requests. This implies that the PPC service could detect clicks occurring by this bot given the improper request ordering.

Fiesta Economic Model

Based on our observations of the Fiesta clickbot and our investigation of the Fiesta PPC service, we believe that we have identified the economic model of both the Fiesta PPC service and the Fiesta clickbot. This model introduces the notion of a click fraud middleman whose job is to abstract ad revenue from the generation of fraudulent traffic. This is a significant change in the economic structure of the click fraud ecosystem that was not present for Clickbot.A.

We suspect that Fiesta PPC has entered into revenue sharing agreements with several advertising sub-syndicates. As part of this agreement, Fiesta PPC operates a search engine that displays the ad sub-syndicate's ads. Each of these ads is routed through the Fiesta Click Server. When an ad click occurs, the revenue is divided between the parties. The Fiesta PPC service then distributes the search engine traffic amongst their ad sub-syndicates through the use of the Fiesta PPC Click Server.

Investigation of the Fiesta PPC website supports these theories. The site contains links to traffic bid information based on region, web forums (which are currently not working), and an affiliate application form.

Inserting a middleman into the click fraud path is an innovative structural development. A PPC service allows botmasters to generate revenue without regard to the specifics or defenses of advertising networks, while simultaneously allowing the middleman service to focus on ad revenue without engaging in traffic generation.

Concurrent with our own work, a separate study discovered a business relationship between the Fiesta PPC service and the operators of the Koobface botnet [142]. This study detailed the economics of Koobface, revealing that the botnet generated some of its revenue by interacting with various affiliate services in exchange for payment. Koobface utilized Fiesta PPC as an affiliate in addition to other PPC services. We believe the Fiesta bot operator has established a similar business relationship with the Fiesta PPC service.

Prevalence

We observed two different PPI services distributing Fiesta bots across a three month timespan. In one instance, a PPI service served the Fiesta bot binary for over 12 hours. Through creative use of the Google search engine combined with our own samples, we were able to locate four different C&C server IP addresses across three different domain names. Since the C&C server used by our bot was hard-coded into the binary and varied between dropper services, we suspect that Fiesta bots released via different mechanisms use different C&C servers. Using the same Google techniques, we have also identified 22 domain names that act as search engines for the Fiesta service, spread across three IP addresses.

7.4 The 7cy Clickbot

Clickbot.A, 7cy and Fiesta differ significantly in their behavior. While Fiesta and Clickbot.A use levels of indirection between organizations to launder clicks, 7cy attempts to evade detection by

```
1 GET /p6.asp?MAC=00-0C-29-24-29-12&Publicer=bigbuy HTTP/1.1
2 Host: in.7cy.net
3 User-Agent: ClickAdsByIE 0.7.4.3
4 Accept-Language: zh-cn,zh;q=0.5
5 Referer: http://in.7cy.net/p6.asp
6 Content-Type: application/x-www-form-urlencoded
7 Connection: Close
```

Figure 7.5: Initial 7cy C&C request. The MAC-based bot ID and user-agent are shown in bold.

```
1 http://housetitleinsurance.com
2 http://www.google.com/url?sa=t&source=web&ct=res&cd=8&url=http://housetitleins...
3 90
4 15
5 CLICK
6 /search/{|||}epl={|||}yt={|||}qs
7 RND
8 5
9 NOSCRIPT
```

Figure 7.6: Excerpt of response from C&C server. Note that whitespace between lines is removed.

simulating human web-browsing behavior. The 7cy C&C language controls the bot's click behavior by specifying an initial site to "surf," a series of page content patterns for identifying desirable links to click on, and an inter-click delay time. The bot then leverages timing by introducing a random amount of jitter into the delay between clicks. Separately, the C&C directs the bot to generate more browsing traffic during popular times such as the evening and the workday. Compared to Fiesta, 7cy requires a substantially different C&C language and surrounding botmaster infrastructure, which we detail next.

C&C Structure

A 7cy bot locates the C&C server by resolving the domain name `in.7cy.net`, and then communicates with that server using HTTP. We show a sample C&C request in Figure 7.5. Line 1 includes the bot's network MAC address, presumably as a unique identifier. Line 3 presents a user-agent specific to the bot family, as well as a version number.

After receiving a request, the C&C server will respond with one of three messages: (*i*) an instruction to wait for a specified time period, (*ii*) an HTTP `302` response redirecting the bot to another C&C server, or (*iii*) specific click fraud instructions. We refer to the latter as an instruction "batch." Each batch is comprised of "jobs," and each job is comprised of "clicks." Within a given batch, each job specifies a different site to target for click fraud, and each click corresponds to an HTML link to visit. Jobs specify web-surfing sessions: their clicks are applied to sites as they result from previous clicks, rather than to the initial site. On average there are 18 jobs per batch and 9 clicks per job.

Figure 7.6 shows an excerpt of one batch. Lines 1 through 4 constitute a header for a single job.

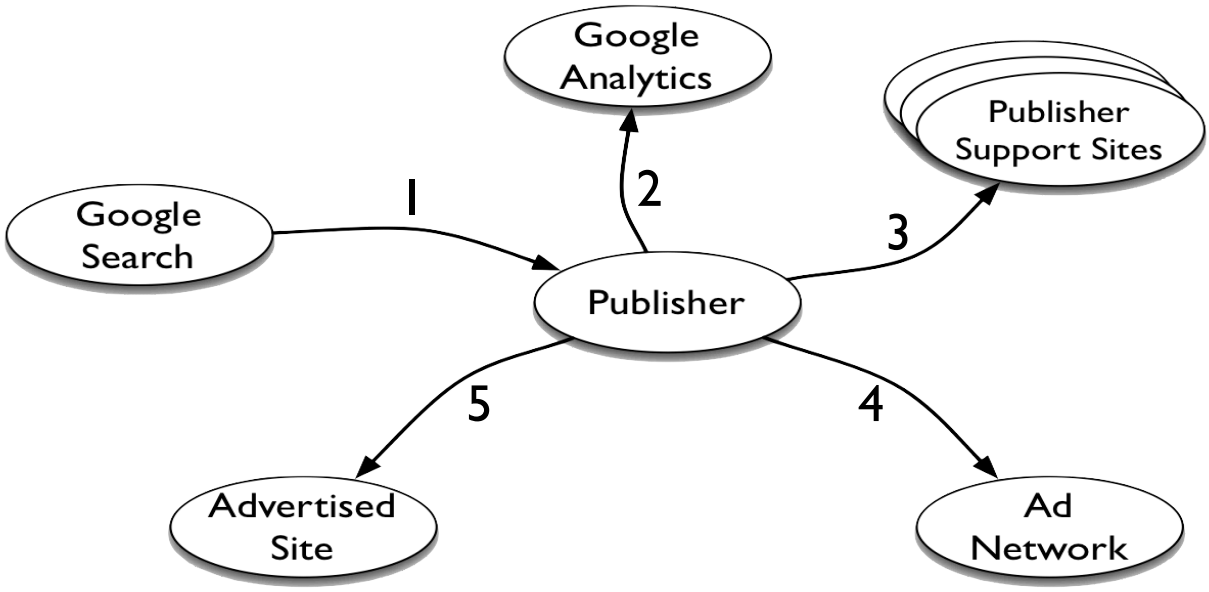


Figure 7.7: General progression of a 7cy "job." Arrows represent HTTP requests and go from the domain of the refer to the domain of the host. Note that the publisher is often a parking page.

Line 1 specifies the website at which the bot should begin the browsing session. Line 2 specifies the referrer to use in the initial request to the target site, although an actual request to the referring site is never made. Line 3 specifies a time limit after which new instructions will be requested if the clicks have not been completed yet. Line 4 specifies the number of clicks in the job. The structure seen in lines 5 through 9 describes a particular click. This structure (with possibly different values) repeats the number of times denoted on line 4 to complete the job. Line 5 specifies the action to perform. Several values other than `CLICK` have been observed, but seem to be ignored by our specimens. Given the rarity of these other commands (less than 0.01% of total commands), we suspect they are erroneous, or a new feature still in testing. Line 6 specifies token patterns to search for on the current page when selecting the next click. Tokens are delimited by the five characters "{|||}". Line 8 specifies a time delay for the bot to wait before performing the next click. Once all clicks in the job are specified, a new job header occurs or the C&C transmission ends.

After receiving a batch of instructions from the C&C server, a bot will begin traversing web pages as described in the instructions. Many of the sites targeted by the bot are hosted at parked domains. Examples include `housetitleinsurance.com`, `quickacting.com`, and `soprts.com`. We call these websites *publishers*. These sites mainly consist of links and ads within the page body, and keywords in the HTML `meta` tag which relate to the theme suggested by the domain name. They may also provide a link to contact the owner and purchase the domain name.

Although the domains and advertising networks vary across jobs, the traffic patterns follow progressions that we can aggregate into a graph, shown in Figure 7.7. Edges correspond to HTTP requests made by the bot. The origin of an edge corresponds to the domain of the referrer used in the request, and destination of an edge corresponds to the host to which the request is made.

```
1 GET / HTTP/1.0
2 Referer: http://www.google.com/url?sa=t&source=web&ct=res&cd=8&url?sa=t&source...
3 User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
4 Host: housetitleinsurance.com
```

```
1 HTTP/1.1 200 OK
2 Cache-Control: no-cache
3 Pragma: no-cache
4 Content-Length: 13776
5 X-AspNet-Version: 4.0.30319
6 Set-Cookie: SessionID=6492595d-c592-419a-bf16-0cad97eef767; path=/
7 Set-Cookie: VisitorID=5f68a43f-6cf3-4a2f-831c-127ce007b646&Exp=11/29/2013 8:38...
8 Set-Cookie: yahooToken=qs=06oENya4ZG1YS6...HO8xG7uL E1uBAe 5qKwGUov0xhAWIvfCJZ1E...
```

Figure 7.8: Selected headers from request (top) and response (bottom) of a publisher's webpage.

The first HTTP request a bot makes in a job is always for the URL of the publisher's website, using a referrer header mimicking a previous Google search (not actually performed) which could plausibly lead the user to the publisher's website (step 1). Next, the bot loads the publisher's website as a browser would, fetching all supporting objects (pictures, scripts, etc.) as dictated by the initial request (steps 2, 3). Once the bot has downloaded the publisher's webpage, it selects an in-page ad matching the search pattern specified via C&C for clicking. If multiple links on the page match the pattern, the bot makes a random selection. Each link on the webpage points to a "trampoline" page at the publisher's site, resulting in HTTP `302` redirects to the ad network and on to the actual advertised site. This behavior allows the publisher and the ad network to detect when the bot clicks on an ad. The bot follows this redirection chain (step 4) and loads the advertised site (step 5). A job often includes several clicks designed to simulate link-clicking behavior on the advertised site.

Specific Fraud Example

In order to demonstrate several details of the traffic produced by a 7cy bot, we now present excerpts of traffic from an actual job. In this job, the publisher is `housetitleinsurance.com`, the ad network is `msn.com`, and the advertised site is `insureme.com`. Figure 7.8 shows the bot's initial request to the publisher's website and the corresponding response. Note that as the bot is now issuing requests to published websites, the `User-Agent` presented in line 3 of the request has been changed to `Mozilla/4.0` rather than `ClickAdsByIE`.

In this instance, after the bot had loaded the publisher's site the bot clicked on a link to the publisher's own domain. This caused the bot to send another series of requests to the publisher as the corresponding site was loaded. Throughout this exchange, we observed that the publisher changed a portion of the cookie originally set on the bot in Figure 7.8 and began including a value similar to the cookie in links included on the page. This is seen in Figures 7.8-7.10 as the bold portion of the cookie changes.

The use of cookies and referrers represents an increase in complexity over the techniques used

```
GET /?ld=4vnjCbJ-GAvwzaNZFHBC2hWDhbZSs2HbnQAVmreNgXqjJdT0CGnrnZiVXS01aPdMH1DdL...
Referer: http://housetitleinsurance.com/online/find/home/owner/find/home/owner...
         ...yt=qs%3d06oENya4ZG1YS6...HO8xG7uLGV-ZMa5qKwGUov0xhAWIvfCJZ1EtVWLOl...
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: 948677.r.msn.com
```

```
HTTP/1.1 302 Object Moved
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Location: http://www.insureme.com/landing.aspx?Refby=614204&Type=home
Set-Cookie: MSConv=4vfcf6a1f935caa89943fce63a4bbf1574fc5c1f28c000945ebcd99d208...
Set-Cookie: MSAnalytics=4v76de0ef30bff74b972b5855ec3be14bc0c26342d22158a9ceaa6...
Content-Length: 202
```

Figure 7.9: Selected request (top) and response (bottom) headers for an advertised site's URL.

```
GET /landing.aspx?Refby=614204&Type=home HTTP/1.0
Referer: http://housetitleinsurance.com/online/find/home/owner/find/home/owner...
         ...yt=qs%3d06oENya4ZG1YS6...HO8xG7uLGV-ZMa5qKwGUov0xhAWIvfCJZ1EtVWLOl...
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: www.insureme.com
```

```
HTTP/1.1 301 Moved Permanently
Cache-Control: no-cache, no-store
Pragma: no-cache
Content-Length: 44447
Location: http://www.insureme.com/home-insurance-quotes.html
Set-Cookie: ASP.NET_SessionId=4u4vxv45vupmetvi2f4ghoqu; path=/; HttpOnly
```

Figure 7.10: Selected request (top) and response (bottom) headers for a request for an advertised site.

by Clickbot.A [37]. Clickbot.A strips the referrer from requests in order to obscure the site at which the traffic originated. While there are plausible reasons for removing the referrer in normal use cases, removing the referrer does introduce a notable characteristic into the traffic. Likewise, cookies help the traffic produced by 7cy to appear more natural and could potentially be used to construct the illusion of a user or browsing session for an ad network.

The bot will ultimately browse away from the publisher's website when a request is issued to the ad network in order to obtain a redirect to the actual website being advertised. This request is shown in Figure 7.9. Note that this request uses a `Referer` which includes a value similar to the `yahooToken` value previously set as a cookie on the bot by the publisher. The changed portion is shown in bold.

Lastly, a request is made to load the actual site being advertised. In this particular case the parking page has been moved within the same domain so a `301` redirect is issued. This is unrelated to the click fraud infrastructure. As shown in Figure 7.10 this request also includes a referrer header which includes the `yahooToken` value used in other requests.

We summarize the pattern of traffic described above in Figure 7.11. Each line corresponds to

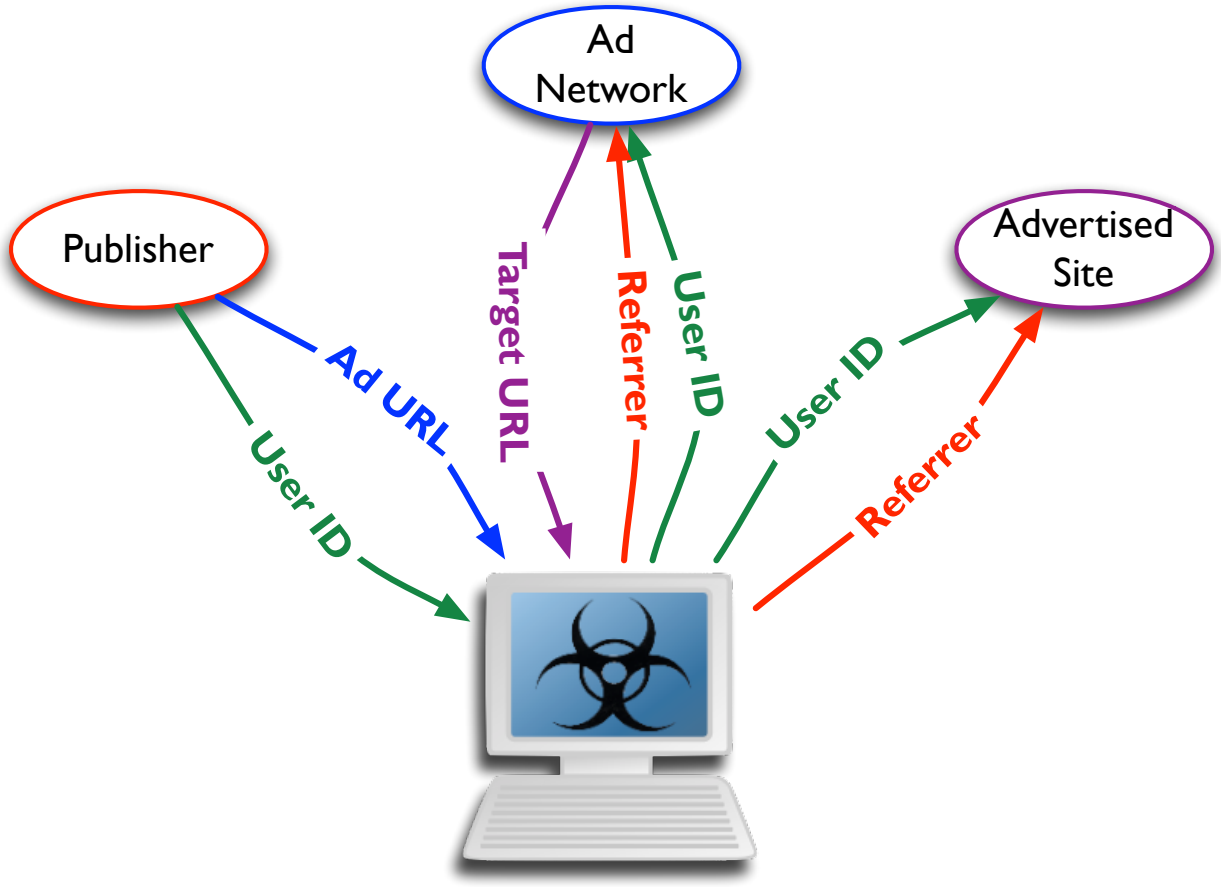


Figure 7.11: Flow of critical information in 7cy click fraud.

a piece of information and each bubble corresponds to a distinct server. The publisher's domain name is included in the referrer to both the ad network and the advertised site. The ad URL is supplied by the publisher and contains the domain of the ad network. The target URL is supplied by the ad network and contains the advertised site domain name. Lastly, the `yahooToken` likely containing a user ID is set by the publisher and given to the ad network and the advertised site.

7cy Economic Model

While the economic structure of 7cy is relatively clear, the parties involved and their roles are not. The pattern of traffic suggests that the advertised site (e.g., `insureme.com`) is paying the ad network (e.g., `msn.com`), which is paying the publisher (e.g., `housetitleinsurance.com`). Additionally, the domain names appear to be registered to multiple distinct parties. Unfortunately, it is unclear whether the publisher is paying the botmaster, or the publisher itself is the botmaster. If the publisher is paying the botmaster then it is unclear exactly how many distinct parties are acting as publishers.

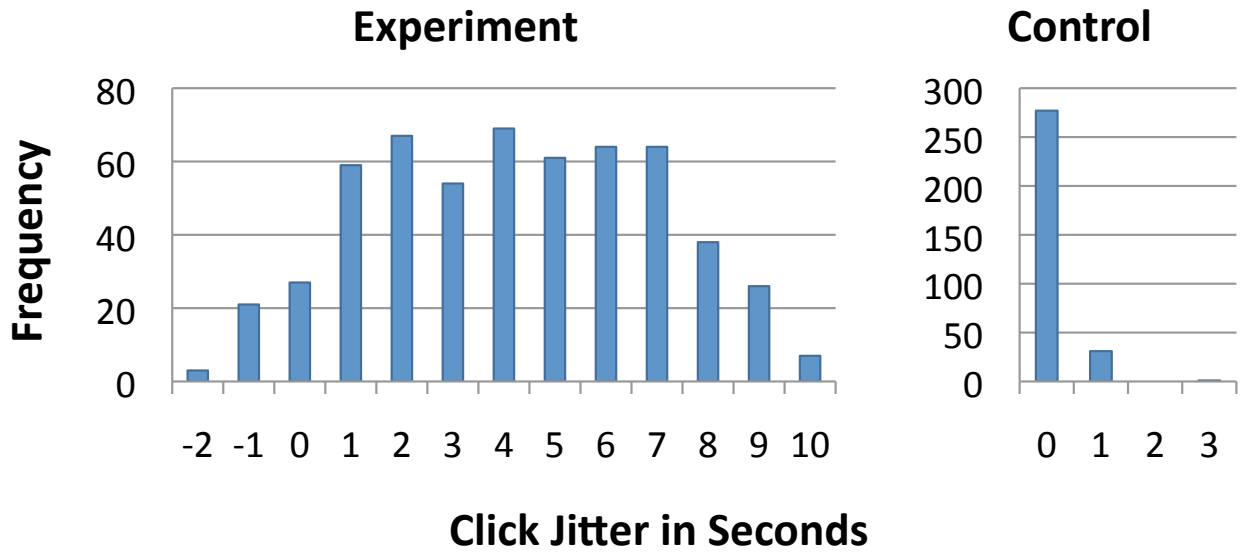
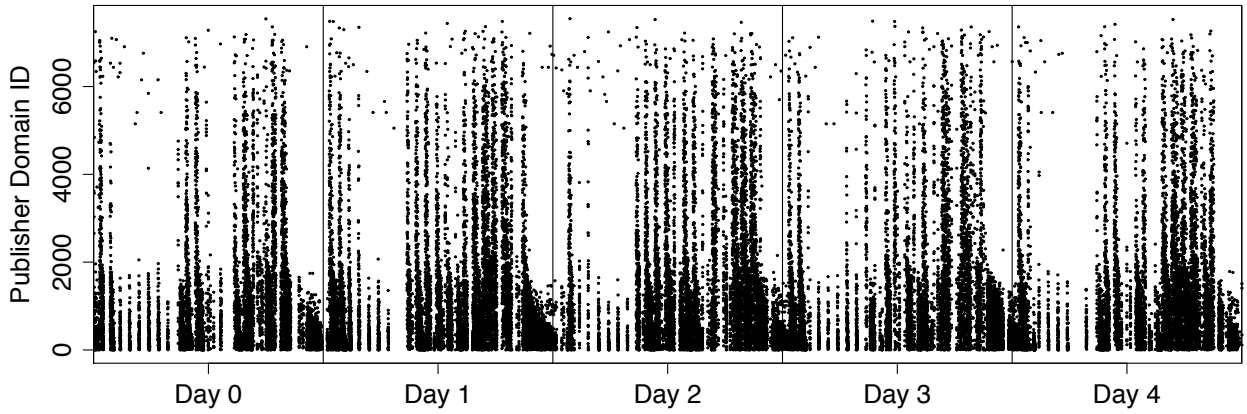


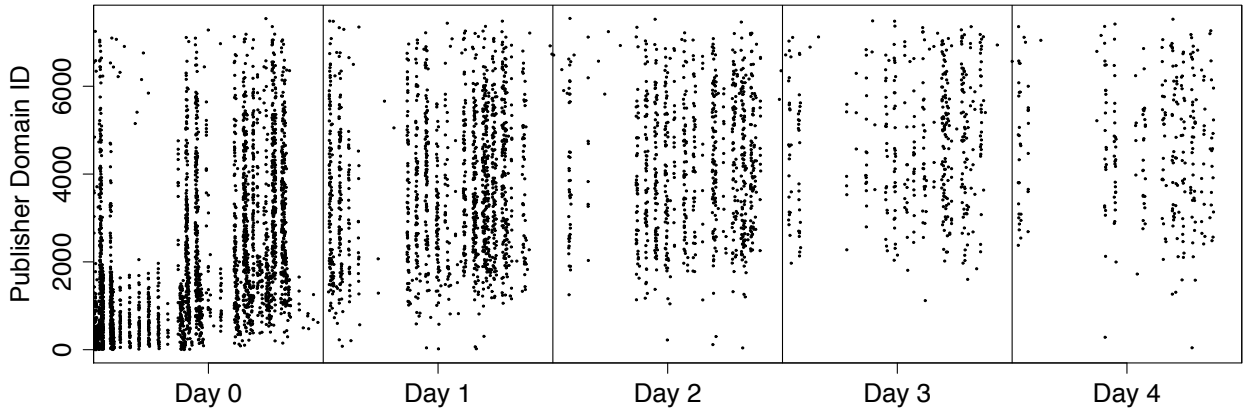
Figure 7.12: Measurement of the amount of jitter introduced into inter-click delay by the clickbot binary and surrounding infrastructure compared to jitter introduced by infrastructure alone.

Timing and Location Specific Behaviors

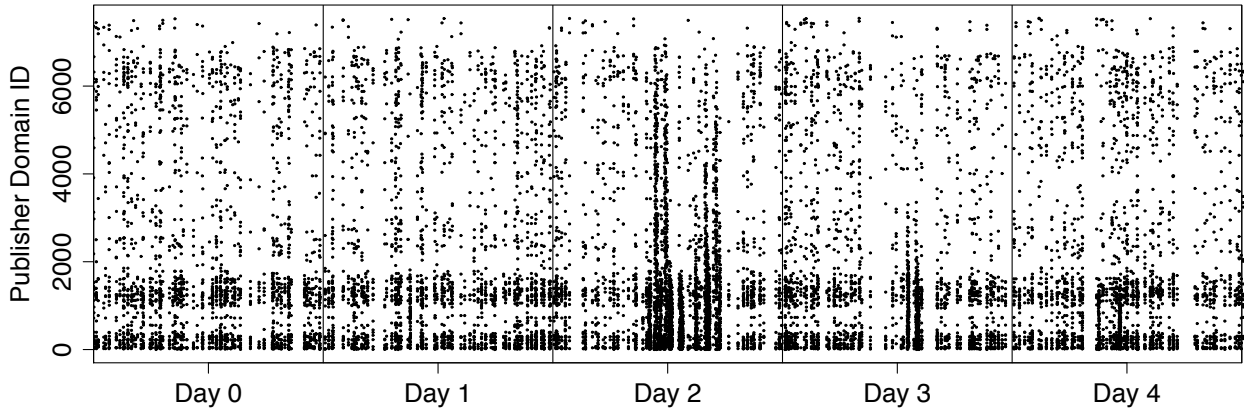
Timing Variance: In order to appear more human, the bot introduces jitter into the delays specified by the C&C language. The results labeled "Experiment" in Figure 7.12 show the differences we observed between the time delay specified in the C&C and the bot's actions in our contained environment. We conducted these measurements by feeding the bot artificial C&C with specific wait times while reflecting all HTTP traffic internally to a sink server within our farm. We then measured the inter-arrival time of requests at that sink. In order to confirm that the jitter observed was the result of the bot's own behavior and not our honeyfarm environment, we also performed a control experiment in which we made HTTP requests at a constant rate from within an inmate VM, and then measured the variance in the same way as with the actual bot. The results labeled "Control" in Figure 7.12 indicate that the jitter introduced by the honeyfarm is infrequent, small, and always positive. Combined, these results show that the 7cy clickbot is introducing both positive and negative variance, on the order of seconds, into the inter-click delay.



(a) Publisher domains seen from milking via the US over all 5 days of the study.



(b) Publisher domains seen from milking via the US over all 5 days of the study, with domains plotted only the first time they occur.



(c) Publisher domains seen from milking via Japan over all 5 days of the study.

Figure 7.13: (Figure continued on next page.)

(d) Publisher domains seen from milking via the US during the 1st 24 hours.

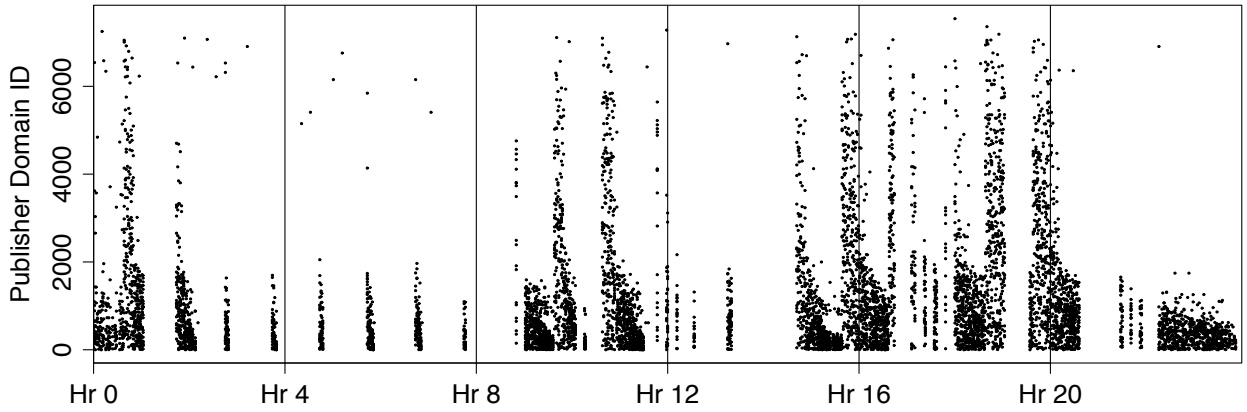


Figure 7.13: (Continued) The plots above show the domains the milker was directed to for click fraud as a function of time. The vertical axis represents all domains observed, ordered from most frequently seen (bottom) to least frequently seen (top). Frequency was defined with respect to the entire study. Note that Day 0 begins at 5am GMT.

Instructions Over Time: In order to gather data about the characteristics of the C&C over time and the influence of bot location on the behavior of control servers, we also conducted a C&C milking study of the 7cy infrastructure. As part of our study we build a milker which connected to 7cy C&C servers via Tor [40] exit nodes in 9 countries throughout North America, Europe and Asia. These countries were Canada (CA), Spain (ES), France (FR), Hong Kong (HK), Japan (JP), South Korea (KR), Russia (RU), Singapore (SG), and the United States (US).

Our milker mimics the behavior of a 7cy bot by contacting the 7cy C&C server and requesting work. We record all network traffic in the exchange, but do not carry out any of the fraud specified by the C&C server. Our milker is implemented entirely in Python and is approximately 370 lines of code. Our study milked the C&C server continuously for five days starting Thursday, January 13 2011, at 5am GMT.

All initial C&C requests, regardless of the Tor exit node, were sent to `in.7cy.net`. This mimicked the behavior we observed in our actual specimens. Recall from Section 7.4 that the C&C server's responses can be classified as "Wait" (delay for a fixed time period), "Moved" (a `302` redirect to another server), or "Batch" (instructions for click fraud). On occasion the C&C server returned a `400`-level error code, empty response, or no response, all of which we classify as "Other." When connecting from Japan, the C&C server occasionally returned a web page which seemed to be under development. We likewise classify this as "Other."

We observe that the C&C servers target some sites for click fraud more often than others. The C&C samples obtained by our milker contained 366,945 jobs directing traffic towards 7,547 unique domains. An analysis of the traffic reveals that although 7,547 unique domains were seen across all countries, 75% of jobs targeted only 1,614 of the domains.

Figure 7.13 shows the domains targeted by the 7cy bots with respect to both country and time. The horizontal axis represents the time in seconds since the start of the milking study. The vertical axis is the sorted target domain ID, where domain 0 is the most targeted domain, and domain 7,546 is targeted least. Figure 7.13a plots the domains sent to our US Tor exit node milker over the 5 day period. The distinctive gaps in the data are the result of successive wait commands given to us by the C&C server. Figure 7.13d is an expanded view of the first day of Figure 7.13a. We see that the server seems to work on an hourly cycle, dispensing a minimum amount of click fraud instructions to the most trafficked sites each hour. The US exit node received more click fraud instructions during peak US Internet use times, such as the work day and evening. In non-peak hours, the US exit node received more wait instructions to generate a lower volume of traffic. Interestingly, all exit nodes except Japan and Korea showed timing patterns in sync with the US despite time zone differences.

Japan and Korea, however, display a pattern similar to each other and distinct from other countries examined. Figure 7.13c shows domains served to the Japanese Tor exit nodes with respect to time, over the entire 5-day milking period. This figure shows the same distinctive bands, however the distances between bands and widths vary. While these countries do appear to have a strong, periodic schedule and do visit some sites considerably more than others, traffic appears to be distributed relatively uniformly throughout the day.

Figure 7.13b shows the same data as Figure 7.13a except all duplicate domains have been removed. This means once a specific domain has been observed at time t, it is no longer plotted for time greater than t. This figure illustrates that although there is a clear periodic pattern to the traffic, the domain names involved vary throughout the observed time. The other countries surveyed have similar behavior.

Beyond differences in timing, the distinct behavior seen in Japan and Korea is also characterized by differences in the C&C instructions received. Table 7.2 depicts the requests and responses made by the milker. Japan and Korea are redirected to `3.95622.com` relatively frequently, although no other countries are directed there. Similarly, Russia, Spain, Germany, Hong Kong, Canada and the United States are redirected to `1.95622.com`, although Japan and Korea are rarely directed to that domain. Only Singapore received no redirects.

In addition to differences in traffic handling and timing, milked C&C revealed a correlation in which domains were served to which countries. In order to determine the degree of correlation between two countries, we calculate for each country the percentage of overlap of its domains to the two countries' combined set of domains. In order to develop a standard of correlation, Table 7.3 shows the result of correlating a randomly selected half of a country's traffic with the remaining half. This provides a standard for determining that two countries are similar. Pairwise analysis of countries revealed that all countries other than Japan and Korea are strongly correlated. These results are presented in more detail in Table 7.4, where we see that Japan and Korea are somewhat correlated with each other and relatively uncorrelated with the rest of the world.

Although there is a strong correlation between domains served and country, the cause for this correlation is not clear as the domains served to Japan and Korea appear similar to domains served to all other countries. The domains which are more common in Japan and Korea do not appear to host content in either Japanese or Korean nor contain ads specific to Japan or Korea. The

Country	Host	Total Req.	Wait	Batch	Moved to 1.95622.com	Moved to 3.95622.com	Other
CA	1.95622.com	193	24	167	0	0	2
	in.7cy.net	2,947	338	2,360	193	0	56
ES	1.95622.com	217	35	178	0	0	4
	in.7cy.net	2,935	327	2,333	216	0	59
FR	1.95622.com	215	18	192	0	0	5
	in.7cy.net	2,883	336	2,252	215	0	80
HK	1.95622.com	323	26	290	0	0	7
	in.7cy.net	2,253	438	1,465	323	0	27
JP	1.95622.com	10	0	10	0	0	0
	3.95622.com	777	378	396	0	0	3
	in.7cy.net	1,656	176	292	10	778	400
KR	1.95622.com	1	1	0	0	0	0
	3.95622.com	1,191	598	590	0	0	3
	in.7cy.net	1,286	22	37	1	1,193	33
RU	1.95622.com	139	14	121	0	0	4
	in.7cy.net	3,520	259	3,048	139	0	74
SG	in.7cy.net	4,238	160	4,000	0	0	78
US	1.95622.com	225	29	194	0	0	2
	in.7cy.net	3,022	322	2,425	225	0	50

Table 7.2: Types of C&C responses received at geographically diverse locations.

	CA	ES	FR	HK	JP	KR	RU	SG	US
Internal Correlation	63.7	61.7	59.8	55.6	43.7	65.2	68.0	74.4	63.0

Table 7.3: Correlation within each country if visits are partitioned randomly in half. Correlation is measured as the percent of domains seen in both halves which were seen in either half.

	CA	ES	FR	HK	JP	KR	RU	SG	US
US	79.5	79.6	78.0	72.5	32.0	12.1	81.0	83.3	100.0
JP	32.2	32.2	31.8	32.7	100.0	42.6	31.5	31.4	32.0
KR	12.4	12.3	12.1	12.3	42.6	100.0	12.0	12.1	12.1

Table 7.4: Correlation of domain names served to various countries. Note that all countries except Japan and Korea are strongly correlated to each other as evidenced by their correlation to the US.

correlation between Japanese and Korean IP addresses and domains served may be related to the ad network being targeted by the bot, rather than the target audience for the content on the domain. We speculate that perhaps some ad networks are more tolerant of or prefer traffic from one country as opposed to others, and so it is more profitable to direct traffic from those countries to those ad networks. As the format of the ad URL is determined by the ad network, this viewpoint is supported by the fact that a similar correlation was seen in tokens to search for in URLs to click on.

The location and time-specific behaviors displayed by 7cy represent a notable departure from the methods of Clickbot.A [37]. 7cy displays time-sensitive behavior both in the randomness introduced to inter-click timings as well as the variations of traffic load with respect to time-of-day. The evidence of location-specific behavior also represents an added degree of complexity over Clickbot.A. These behaviors make bot-generated traffic appear more realistic and represent an advance in emulating human behavior.

7.5 Discussion and Summary

In this chapter we presented an in-depth analysis of two distinct families of clickbots, Fiesta and 7cy, deriving extensive behavioral information about these families. This allowed us to establish a profile of the capabilities of the bots as well as the economic motives and incentives of the parties involved. Utilizing these insights into bot behavior and the structure of click fraud systems, we are now able to discuss potential techniques for defenses and safeguards against bot-generated traffic.

Through our study of the Fiesta clickbot we have described a click fraud model in which a service provider acts as a middleman for fraudulent traffic. The middleman pays money for generated traffic, and generates his own revenue through agreements with ad sub-syndicates. This previously undescribed approach could allow advances in traffic generation to be abstracted away from advances in hiding fraudulent clicks, potentially driving click fraud innovation.

Studying the 7cy clickbot allowed us to observe the specific mechanisms employed by a clickbot attempting to mimic the behavior of a human. The C&C protocol allows the botmaster to dictate or sell traffic at a fine granularity. We observed the attempt to simulate human-like behaviors, including random browsing of the advertised site and randomized inter-click delays. By milking 366,945 click fraud instructions from the C&C servers via IP addresses in 9 countries we were able to study this botnet's click fraud in detail and discover region-specific behavior.

Having described the behavior of both the Fiesta and 7cy clickbots, we would like to offer a brief discussion of potential techniques for detecting bot-generated traffic. One approach sites could employ is to develop a set of features characteristic of legitimate traffic and flag browsing sessions which appear abnormal according to these features. Features which would address the bots seen in this chapter include the user's mouse movements on the page and the depth of browsing through the site, as these bots differ significantly in these regards from a typical human. Advertised sites may further this technique by correlating atypical features with the domain name of the referrer, and in doing so build a list of publisher domains in use by botmasters. Another conceivable detector is the invisible embedding of HTML links into a site's pages. Any visitor

clicking such "honeylinks" is likely to be a bot. In addition to the above suggestions for potential detection techniques, we have pursued collaboration with industry to enhance bot detection.

We build on this work in Chapters 8, 9, and 10, exploring the ecosystem of large-scale click fraud botnets and their supporting ecosystem further.

Chapter 8

ZeroAccess: Background and Evolution

ZeroAccess was a vast and complex peer-to-peer (P2P) botnet that served as a delivery platform to distribute a variety of malware modules over its lifetime, each with its unique command-and-control (C&C) and monetization strategy [17].

8.1 ZeroAccess Evolution and Takedown

ZeroAccess was a complex botnet that has undergone several stages of evolution, which we recount here. Although first described solely as a "rootkit," ZeroAccess developed into a vast *peer-to-peer* (P2P) botnet and malware delivery platform.

Early Life, 2009–2011

Initial reports of the "ZeroAccess rootkit" date to 2009 [56]. In 2010, the InfoSec Institute's detailed analysis described it as a "platform to deliver malicious software" [17]. At this stage, the main malware delivered using ZeroAccess was "FakeAV",[1] with an estimated 250,000 computers infected.

First generation P2P Botnet, 2011–2013

In May 2011 a radically new version of ZeroAccess emerged [109]. In this iteration, ZeroAccess retained its kernel-mode rootkit components, but changed both its communication and monetization strategies. This version spread itself via exploit packs (e.g., BlackHole [68]) and social engineering [64, 150].

The defining feature of this iteration of the botnet was the introduction of a decentralized, TCP-based P2P communication protocol. The protocol used cryptography and obfuscation as well as other common P2P features such as "supernodes" that served to orchestrate large portions of the

[1] FakeAV is malware that claims to be anti-virus software to extort users into paying money to remove fictitious infections.

network's activity. The network allowed the botmasters to maintain decentralized control while relaying commands and payloads to infected computers worldwide [149].

The P2P protocol included cryptographic signing of malicious payloads, which hardened the botnet against attempted hijacking by preventing untrustworthy peers in the botnet from successfully delivering payloads other than those cryptographically signed by the actual botmaster [149].

In addition, the monetization strategy changed with this generation. ZeroAccess moved away from FakeAV payloads and instead began distributing Bitcoin miners and click fraud modules.[2] From a technical perspective, the primary click fraud malware used in this era operated in the indiscriminate "auto-clicking" fashion we describe in Section 9.4.

Alongside the click fraud and Bitcoin payloads, ZeroAccess itself was also sold as a service on underground forums [109], enabling cyber-criminals to use the ZeroAccess rootkit to distribute their own malicious payloads.

This iteration of the botnet also saw an increase in botnet population. At the height of infections in early 2012, estimates placed the botnet population at over 500,000 [100]. Despite the age of the botnet and its subsequent evolution, as of August 2013 there were still over 30,000 computers infected with this generation [109].

Second generation P2P Botnet, 2012–2013

In July 2012, ZeroAccess evolved into the form predominant as of November 2013. According to Symantec, by August 2013, this generation had an estimated population of over 1.9 million computers [109]. This iteration included several changes to the malware structure, the protocol, and the payloads. The most distinguishing change to ZeroAccess in this era was a move away from the kernel-mode rootkit component, with all of its functionality now replicated in user-space [109]. Other changes include a move to UDP from TCP for the P2P protocol (likely to improve network performance), and minor changes in the protocol itself.

The monetization strategy also evolved. This version saw the introduction and massive distribution of a new click fraud payload performing *search-hijacking* (linked to the MagicTraffic click fraud affiliate program [79]), which we discuss in Section 9.5.

Takedown, Late 2013

On December 5, 2013, Microsoft's Digital Crimes Unit and Europol orchestrated a takedown of the Serpent and auto-clicking C&C servers [90, 104], using our technical report [118, Chapter 9] as Exihbit 1 [104]. Since both click fraud modules had a centralized C&C server, a simple seizure of these machines was able to temporarily disrupt click fraud activity. Since the ZeroAccess P2P substrate was still intact the perpetrators were able to distribute an updated module with new C&C IP addresses within hours and fraud resumed. For reasons unknown, the following day the malware authors distributed a new set of modules that halted all click fraud activity but left the P2P

[2] ZeroAccess's shift away from FakeAV occurred just before a major takedown that resulted in the closure of most FakeAV programs [88].

network intact. Inspection of these modules revealed the ASCII text "WHITE FLAG" in apparent surrender [63].

8.2 Technical Background

P2P communication

The P2P C&C substrate of ZeroAccess functions solely to deliver modules to infected machines. This P2P protocol is well understood, has common P2P properties such as peer lists and supernodes [109, 126, 149], and allows ZeroAccess infected computers to communicate with each other directly without the need for a centralized C&C server. Without payloads, ZeroAccess infections simply maintain their P2P membership and functionality, but perform no malicious actions on their own.

Infected ZeroAccess machines generate UDP P2P traffic on four well-known ports. The protocol is constructed in such a way that super nodes in the network have a vantage point of a large portion of the infected population. ZeroAccess has distributed several modules with various monetization strategies over its five-year lifespan. In this work we focus on the two most recent modules distributed throughout 2013. Both these modules perform click fraud and both have a separate HTTP-based C&C channel distinct from the P2P network.

ZeroAccess Modules

During the life of our study ZeroAccess utilized two primary monetization strategies, each using a different module: The "auto-clicking" module (classic click fraud), and the search-hijacking module, which we have named Serpent. Chapter 9 performs an in-depth technical analysis of each of these modules beyond the following summary.

Auto-Clicking Module

The ZeroAccess auto-clicking module performs click fraud by simulating normal Web browser behavior of a user clicking an ad. These clicks occur rapidly, require no user participation, and are not visible to the user of the infected computer. Some users, however, may be alerted to the presence of this module by the increased network activity; in one instance, we observed about 50 MB of click traffic per hour.

The auto-clicking module invokes an actual client Web browser, enabling realistic Web browsing behavior including proper handling of HTML, CSS, JavaScript as well as browser-specific quirks. The module periodically contacts the auto-clicking C&C to fetch a list of publisher Web sites for the bot to visit. It navigates to the URL provided by the C&C in a hidden window, locates the ad on the page, and simulates a user clicking on the ad by requesting the advertiser's URL. The browser faithfully follows the sequence of redirects from the URL until it loads the advertiser site. The auto-clicking module then closes the hidden window and starts anew with the next publisher

in the list provided by the C&C. Notably, the auto-clicking module does not simulate any user actions on the advertiser page and thus does not trigger any conversions.

Serpent Module

The Serpent module interposes on a user's normal interaction with various search engines to redirect the unwitting user to an ad. Whereas the auto-clicking module simulates a user, Serpent sends a real user to the advertiser.

The Serpent module includes a browser component that lays dormant until the user performs a search on one of many search engines. The module silently sends a copy of the search query to the Serpent C&C. The C&C responds with a set of URLs that correspond to ad click URLs for ads related to the search query. Meanwhile, the browser renders the original search result page as the user expects. When the user clicks on a search result Serpent intercepts the click and prevents the browser from navigating to the site intended by the user. Instead, the module redirects the browser to one of the URLs it received from the C&C, in effect creating the appearance of the user having clicked an ad after performing a search on the publisher's site (as opposed to the site on which the user actually performed the search). If the user does not notice having been unwittingly redirected to the advertiser's site (a different site than intended), the user may continue browsing as normal. Since the advertiser site is relevant to the user's search query, the user may even convert. Note that since all Serpent activity is gated on legitimate user activity, this type of click fraud attack is extremely hard for an ad network to detect.

Parallel to this search-hijacking activity, Serpent also used a separate C&C protocol to perform a type of auto-clicking. This auto-clicking protocol and ad network structure were separate from the original auto-clicking module, although the auto-clicking behavior is largely similar.

DNS Queries

Serpent ships with a list of C&C IP addresses hard-coded into each module. These C&C IP's are encoded inside ASCII strings that appear to be domain names ending in `.com`, but are in fact an obfuscated encoding of the numeric IP address. For each Serpent C&C function the malware selects one of these pseudo-domains, decodes the domain into an IP address, and then initiates a connection to that IP. Note that the malware does not need to perform a DNS query since the IP address is encoded in the domain name itself. However, for some unknown reason, the malware authors coded the module to perform a DNS query via Google's public DNS server each time it decodes the pseudo-domain—resulting in a DNS query for each C&C operation—and discards the resulting DNS response (if any).

Most of the ZA Serpent pseudo-domains were not registered prior to our study. For this study we registered all available pseudo-domains, and by observing these DNS queries we are able infer operations the malware is performing. Section 10.2 discusses this DNS data, and the inferences it enables, in more detail.

Chapter 9

The ZA Auto-Clicking and Search-Hijacking Malware

9.1 Introduction

As one of the largest click fraud botnets ever in existence, ZeroAccess's operations are of unique interest in understanding how mass click fraud campaigns are perpetrated. Much of ZeroAccess has been well studied, including the infection vector and the peer-to-peer (P2P) command-and-control (C&C) protocol [56, 109, 149, 150], and several reports have identified its use of click fraud [109, 149]. However, we are unaware of public work documenting the click fraud process in technical depth.

In this chapter we focus on documenting the click fraud behavior of the ZeroAccess botnet and the infrastructure it uses in pursuit of this goal. We take a multifaceted approach, including a combination of binary and network analysis, malware execution, and direct interaction with the botnet C&C servers. In particular, we describe how ZeroAccess uses two different "modules" to carry out distinct forms of click fraud: auto-clicking and search-hijacking.

Auto-Clicking: This ZeroAccess module automatically clicks on advertisements sent via the module's C&C. These clicks occur rapidly, unseen by the user and independent of any user interaction. Section 9.4 describes the behavior and infrastructure of the ZeroAccess auto-clicking module.

Search-Hijacking: This ZeroAccess module, which we have named Serpent, fetches ads relating to *real search queries* generated by the user on the infected machine. When the user clicks on a search result, the module intercepts the click and instead performs a separate fraudulent ad click related to that search query. It then redirects the user to an advertiser's Web site. Given the user interaction and the click's relation to the search query, such fraud may lead to advertising conversion,[1] resulting in higher revenue for the criminals. We discuss this process, and the associated

[1] In advertising, a conversion is a click that leads to some user interaction on the advertiser's Web page. What constitutes a conversion can vary based on advertiser. Such clicks are said to be "high quality" because of the user interaction.

MD5	*Last Obtained*
Auto-Clicking module	
`51ba6261e44c60b2f891fabfaa47d0ad`	Nov. 22, 2013
Search-Hijacking module	
`7128a957f5c9c9a69385f5332ca6338c`	Nov. 22, 2013
`3aec103d38c7520229e18af260c5a00d`	Sep. 26, 2013
`36616e8f309b35f8e090068690272239`	June 14, 2013
`8fa08c59e4d205e514f8a978678ba798`	May 30, 2013

Table 9.1: Auto-Clicking fraud and Search-Hijacking module executables used in the analysis.

mechanisms used to achieve it, in greater depth in Section 9.5.

The remainder of this chapter first explains the mechanics of Internet advertising and click fraud, the history of ZeroAccess, and our measurement methodology. We then describe the basic structure of the ZeroAccess malware distribution platform and provide a detailed description of each of its two click fraud modules. In particular we document how each click fraud module uses its own C&C network (also distinct from that used by the ZeroAccess platform) with well over a dozen server IP addresses implicated in our analysis.

This chapter is based on work that appeared as a technical report [118] and served a Exhibit 1 [104] in a legal action against the criminal operators of the botnet.

9.2 Methodology

In this section we describe our malware execution environment, manual analysis techniques, and the ZeroAccess modules we examine.

Collecting Module Sample Executables

Table 9.1 lists the modules used in our analysis of the click fraud modules. We obtained our samples of ZeroAccess by searching malware repositories for traffic patterns consistent with ZeroAccess command-and-control (C&C) behavior and executing each binary in our execution environment (Section 9.2). We have identified thousands of binaries with ZeroAccess C&C behavior found in October and November, 2013.

During execution, ZeroAccess retrieves the auto-clicking and search-hijacking modules for execution. While ZeroAccess transfers modules in an encrypted form, using reverse engineered decryption routines [149] we built a tool that automatically extracts modules from network traces.

Monitored Execution Environment

We execute each binary in a virtualized environment provided by the GQ honeyfarm [91], which supports monitoring malware execution while providing a flexible network containment policy. We use Windows XP Service Pack 3 for all executions. The system can process thousands of binaries per day.

In our experiments the execution environment allows ZeroAccess C&C P2P (UDP) traffic. For all executions we forward HTTP traffic to the intended destination and redirect all other non-C&C TCP traffic to internal sinks. For DNS, our service can answer all queries, even requests without a valid answer or directed at external DNS servers. This feature ensures that domain takedowns during our analysis have limited impact on malware execution. The configurable nature of the DNS server behavior enables us to test ZeroAccess samples with and without DNS resolution. For all other protocol types we provide a sink that will accept packets but does not respond.

In addition to network monitoring, our system collects operating system events, including process creation, file modifications, and registry changes.

Binary Analysis

For static binary analysis we use IDA Pro 6.4 with the Hexrays decompiler.[2] ZeroAccess distributes modules as standard Windows DLLs, a file format natively supported by IDA such that it can disassemble and decompile the modules with Hexrays. We use static binary analysis to obtain the encryption (and decryption) algorithms, domains, and other C&C protocol information for the ZeroAccess modules.

Milking

A *milker* is a program that speaks a particular botnet's C&C protocol and mimics the communications of that malware. Through the use of a milker, we can query information and commands at a much larger scale, with finer granularity, and across more diverse geographic regions, than with traditional malware executions. This technique also allows us to probe specific protocol behaviors in a way that directly executing the malware might not manifest, and to obtain C&C commands without potentially dangerous malware side effects.

For this work we created a milker for the ZeroAccess search-hijacking module's C&C protocol and used it to interact with the module's C&C servers. The milker queries the C&C server and retrieves a list of ads to click on, then simulates a click on one of the results using a headless Web browser. The browser follows redirects, executes JavaScript, and in general is designed to perform similarly to a victim's browser. We ran our milker over several days and describe some of the data gathered in detail in Section 9.5.

[2] `https://www.hex-rays.com/products/ida/index.shtml`

9.3 The ZeroAccess Platform

In this section we describe the base ZeroAccess platform: the botnet software responsible for coordinating communication among very large numbers (millions) of infected computers around the world.

Infection

The first step in a ZeroAccess victim's lifecycle is becoming infected with ZeroAccess. Like many other malware families, ZeroAccess is distributed in a variety of ways, such as drive-by download, social engineering, and pirated software [64]. Each distribution vector results in the installation of software that participates in the ZeroAccess C&C.

An example of this process we have observed begins when a victim browses the Web and inadvertently visits a compromised Web site hosting an exploit kit. The exploit kit detects the victim's browser version and delivers an exploit payload. The payload has two functions: 1) exploit a browser vulnerability, and 2) deliver malware. Upon successful browser exploitation, the payload downloads and executes a ZeroAccess binary. Once executed, ZeroAccess has control of the victim's computer and begins to communicate using the P2P C&C protocol.

Command and Control

The ZeroAccess platform uses a P2P protocol for its C&C, with the primary function of distributing modules and performing updates. The P2P protocol, described in greater detail in other reports [109,149], supports the promotion of a member to a super node. As described by Neville et al., super nodes store ZeroAccess modules and provide them to other nodes upon request [109]. In addition to distributing modules to newly infected hosts, super nodes also host new versions of modules when updated. It is important to note that aside from distributing the click fraud modules, the P2P C&C protocol does not play a role in the execution of click fraud.

Once a victim has been infected with ZeroAccess, it begins by bootstrapping the P2P protocol using a peer list embedded in the binary [109]. The P2P protocol discovers new peers, updates the peer list, and adds itself to the peer list for new nodes to contact. Once the newly infected machine joins the ZeroAccess P2P network, it begins to download modules as instructed by other peers in the network. A super node hosts the modules, and the malware issues a download request to fetch and then execute the module. This process occurs shortly after infection and results in a click fraud module download. Once that module executes, the victim's computer begins to carry out click fraud using a separate C&C protocol as described in Sections 9.4 and 9.5. When an update to the click fraud modules becomes available on the P2P network, the victim learns of the update from one of its peers and contacts a super node to retrieve the latest version.

9.4 The Auto-Clicking Module

The ZeroAccess auto-clicking module performs click fraud by simulating normal Web browser behavior of a user clicking on a Web advertisement. This activity requires no user participation, and is not visible to the user.[3] We present the behavorial analysis of this module, treating the module itself as a black box and observing its contained execution. We observed it performing about one click every two minutes. To evade detection, these clicks were spread across multiple ad networks.

Behavior

The function of the auto-clicking module is to simulate a user "click" on an advertisement. Figure 9.1 shows the operation of the module. The module begins by contacting one of its command-and-control (CF-C&C) servers with a request for click fraud jobs (Step ❶). The C&C server returns a scrambled payload containing a list of "click" jobs. Each job is identified by a host name, a first hop URL and the HTTP `Referer`[4] URL. The module then issues an HTTP request to another CF-C&C server, setting the `Host` header value as specified by the job. This server then redirects the request via an HTTP 303 redirect to a URL, the same URL as in the job (Step ❷). This forms the first hop in the redirection chain. The module then retrieves the URL to which it is redirected, setting the `Referer` header as given in the job (Step ❸). The redirect chain continues normally from this point on, and after a series of redirects the bot fetches the ad URL, at which time the advertiser gets charged (Step ❹).

There is no visible activity in the foreground, so it is difficult for a user to detect that their computer is performing click fraud in the background. This same process repeats multiple times.

Command and Control

This section describes how the bot communicates with the CF-C&C servers to fetch click jobs. Table 9.2 lists the IP addresses of the CF-C&C servers that we observed the bot contacting for fetching commands. Note that these change over time.

The bot contacts one of the CF-C&C servers over TCP port 12757, and sends an obfuscated message (message string XORed by 0x72) identifying the browser User Agent string. In response, the CF-C&C server sends a response (also obfuscated) that contains the following: a domain name, list of first hop URLs to be contacted and a set of `Referer` headers to be set.

After receiving the first hop URLs, the auto-clicking module does not fetch the first hop URLs directly. Instead, it first contacts one of the other CF-C&C servers over HTTP port 80, with the `Host` header set to the domain name provided earlier. Presumably this is an authentication mechanism; earlier versions of the module exhibited similar behavior of not fetching the URLs

[3] Some users, however, may be alerted to the presence of this module by the increased network activity; in one instance, we observed about 50 MB of network traffic per hour.

[4] An HTTP `Referer` header provides the server with the URL of the referring page, that is, the page that purportedy contained the URL being requested.

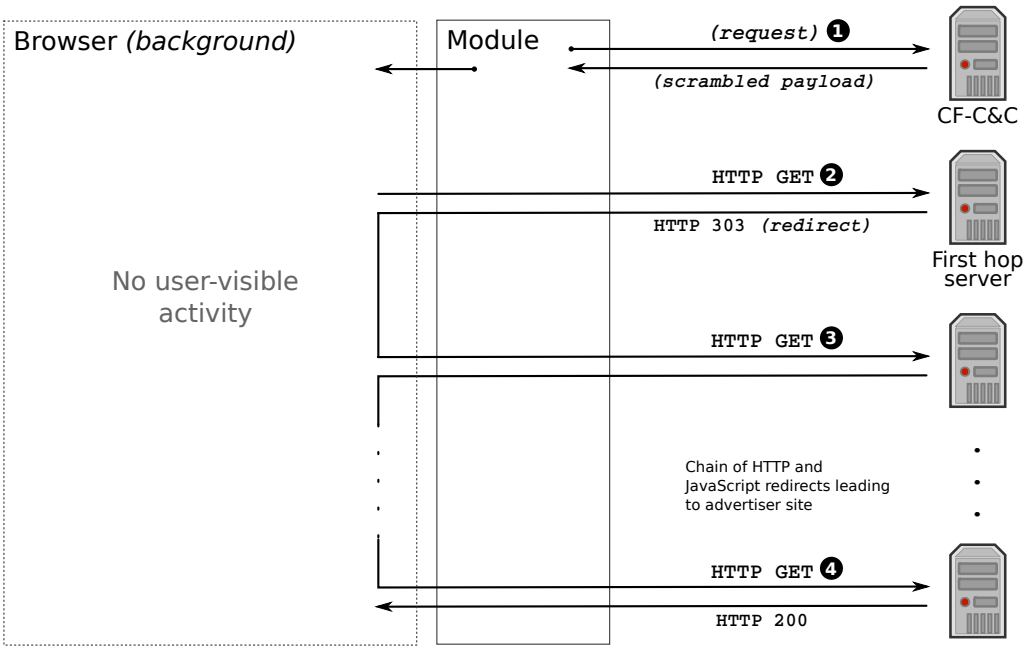


Figure 9.1: Behavior of the auto-clicking module. The module begins by retrieving a list of "click" jobs from its C&C server (Step ❶). For each job, it uses the system browser to retrieve the URL (Step ❷) and follows HTTP and JavaScript redirects (Steps ❸ and ❹).

IP Address	*Observed Date*	*Location*
94.242.195.162	21 Nov 2013	Luxembourg
94.242.195.163	21 Nov 2013	Luxembourg
94.242.195.164	21 Nov 2013	Luxembourg
81.17.18.18	21 Nov 2013	Switzerland
81.17.26.189	21 Nov 2013	Switzerland
46.19.137.19	21 Nov 2013	Switzerland

Table 9.2: IP addresses of C&C servers observed for the auto-clicking module in Table 9.1. The *Obeserved Date* gives the date on which we observed communication between the module and these servers. Location is based on MAXMIND [98] GeoIP service.

Hop	*URL*	*Status Code*	*Notes*
1	`http://46.19.137.19...`	303	Host name set to cvmrpznw.cm
2	`http://[...].traffiliator.com...`	302	Referer spoofed
3	`http://unlimiclick.com/bd...`	200	
4	`http://ads.clicksor.cn...`	200	
5	`http://poomedia.com/ad...`	200	Loaded in iframe
6	`http://us.ad2mi.com...`	302	Loaded in 1x1 pixel iframe
7	`http://searchists.com/search...`	200	JavaScript redirect
8	`http://searchists.com/click/...`	302	
9	`http://click.local.com...`	302	
10	`http://1389.r.msn.com...`	302	Ad URL fetch, advertiser charged
11	Advertiser	200	

Table 9.3: Example redirection chain corresponding to an auto-clicking module "click" from November 21, 2013. [...] is a unique number, in this case 1556987547.

directly [149]. In response, the second CF-C&C server sends an HTTP 303 response code, redirecting the browser to one of the URLs in the list. At this point, the auto-clicking module inserts a supplied `Referer` header and a click chain begins. After a series of redirects, the ad URL gets fetched, resulting in the advertiser getting defrauded.

Example redirect chain

Table 9.3 shows a redirect chain generated by the auto-clicking module, starting from when the bot contacts the C&C server to authenticate itself by setting the `Host` header. In response, the CF-C&C server at `46.19.137.19` redirects the bot to `1556987547.traffiliator.com`. This URL was present in the original click fraud job list, and the bot now inserts the corresponding referrer before fetching the URL, which eventually causes a banner to be fetched in an iframe from `poomedia.com`. In addition to the banner, `poomedia.com` also populates the banner iframe with a 1x1 pixel iframe. This second iframe is loaded with ad URLs and JavaScript code that automatically fetches one of the links at random. This step results in an ad click, which is eventually redirected through other publishers to the ad network, and eventually to an advertiser.

Entities

Depending on the syndication arrangement between different parties in a redirection chain, all of them stand to gain from a fraudulent click that an advertiser pays for, and thus any one of the publishers may be working with the botnet. Over time, from our observations and others [149], different versions of this module have been seen to defraud all major CPC ad networks, including AdCenter, AdWords, 7Search, affinity, and adsimilate. We speculate that this module evades ad network detection by spreading click fraud across a large number of ad networks to hide the high volume of click fraud performed.

Since this click fraud is invisible to the end user (unlike the search-hijacking click fraud module), the user is unlikely to convert. Given that, the use of *smart pricing* (cf. Section 6.2) should in theory discount such malware-driven clicks. However, because of the large number of hops in the syndication chain and the JavaScript redirects that hide the true length or source of the origin of the traffic, it becomes extremely difficult for an ad network to identify that the traffic is being driven by a malware source, as the click fraud traffic mixes in with other legitimate (and converting) traffic from its known syndicators, thus undermining the use of smart-pricing.

9.5 Serpent: The Search-Hijacking Module

The search-hijacking module interposes on a user's normal interaction with various search engines in order to redirect the user to an advertisement that generates the botmaster revenue. Such *search-hijacking* represents a more sophisticated type of click fraud. Whereas the auto-clicking module simulates a real user, the search-hijacking sends a real user to the advertiser. Because the advertiser's site is relevant to the user's search query, the user may in fact interact with the advertiser's site and trigger a conversion, as described in Section 8. We describe the module's search-hijacking behavior in more detail next, and then report on our analysis of this module.

Behavior

Once loaded, the module monitors the the interaction between the user on an infected PC and the browser, waiting for the user to issue a search query to a search engine (Step ❶ in Figure 9.2). We have confirmed that the module recognizes and hijacks Web searches performed using Google, Bing, Yahoo, Ask, and ICQ Search. The module captures the query terms, while allowing the query to go through to the intended search engine (Step ❷). At the same time, the query terms are sent to the search-hijacking module's C&C (SH-C&C) server to retrieve a list of ad URLs to be used for later hijacking (Step ❸). When the user clicks on a search or ad result (Step ❹), the normal click is *hijacked* and the intended URL is replaced with the replacement ad URL retrieved from the SH-C&C server (Step ❺). The browser then fetches and renders the replacement URL instead of the intended search result URL (Step ❻). Although not shown in the figure, retrieving the replacement URL may involve a chain of HTTP and JavaScript redirects. Table 9.4 gives an example redirect chain of a click issued by the module.

The replacement and redirection process operates invisibly to the user. An unsuspecting user will believe that the resulting page corresponds to the search result or ad on which the user clicked on the search result page. It is important to note that neither the advertiser nor the ad network used in the hijacked click may be aware that search-hijacking took place. From their point of view, a hijacked user appears no different from a user arriving via normal search syndication.

Unlike traditional click fraud, such search-hijacking actually delivers a user to the advertiser. Such a user may interact with the advertiser's site and even convert, as described in Section 6.2. Because some fraction of the users will convert, smart pricing may treat traffic from the affiliate engaged in search-hijacking as legitimate and pay for each click.

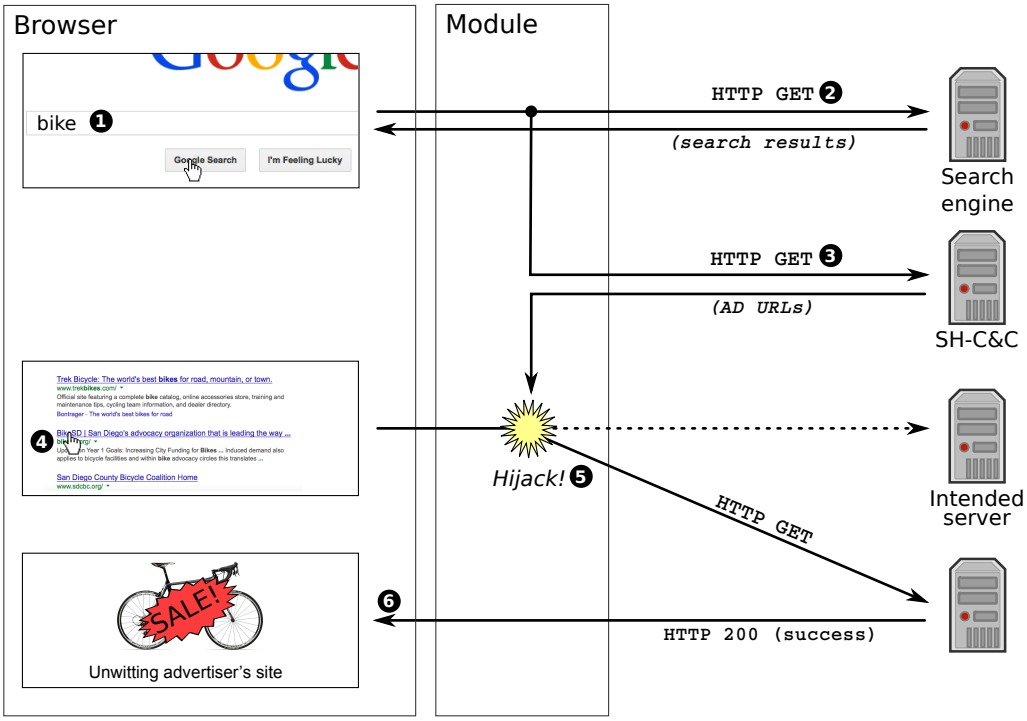


Figure 9.2: Behavior of the search-hijacking module. **Step ❶**: A user enters a term into a search engine. In this example, the user searches for "bike". **Step ❷**: The user's browser performs an `HTTP GET` for that term. In response, the user is presented with the unaltered search result from the search provider. **Step ❸**: In parallel to the user search, ZeroAccess sends the search term ("bike") to the ZeroAccess SH-C&C server. **Step ❹**: The user clicks on a search result on the unaltered results page. **Step ❺**: ZeroAccess intercepts the click. Rather than going to the intended click destination, the user is sent to one of the ad URLs supplied by the ZeroAccess SH-C&C in Step ❸. **Step ❻**: The user's browser displays the result of the ad click, an advertising landing page related to their original query ("bike").

Command and Control

The search-hijacking module's command-and-control (SH-C&C) protocol uses HTTP with a hard-coded set of server addresses. We now detail the different elements of this protocol.

Commands

The primary purpose of the SH-C&C channel is to retrieve a list of replacement URLs to which the user will be redirected when clicking on a query result. When the user performs a search engine query, the module makes an HTTP GET request to a SH-C&C server (Step ❸ in Figure 9.2). The HTTP GET request string is formed as shown in Figure 9.3. The user search terms, together with additional parameters is first formatted using standard URL parameter encoding, using the printf-style format string:

Hop	*URL*	*Status Code*
1	`217.23.3.223/...`	302
2	`http://feed.hype-ads.com/...`	302
3	`http://search.freshcouponcode.com/search.php...`	200
4	`http://c.freshcouponcode.com/redrct.php...`	200
5	`http://c.freshcouponcode.com/click.php...`	301
6	`http://nn.xdirectx.com/clicklink.php...`	302
7	`http://2478799.r.msn.com/...`	302
8	Advertiser	200

Table 9.4: Example of a redirect chain corresponding to a "click" issued by the search-hijacking module on November 14, 2013. Non-final hops with 200-level status codes trigger Javascript or Flash-type redirects.

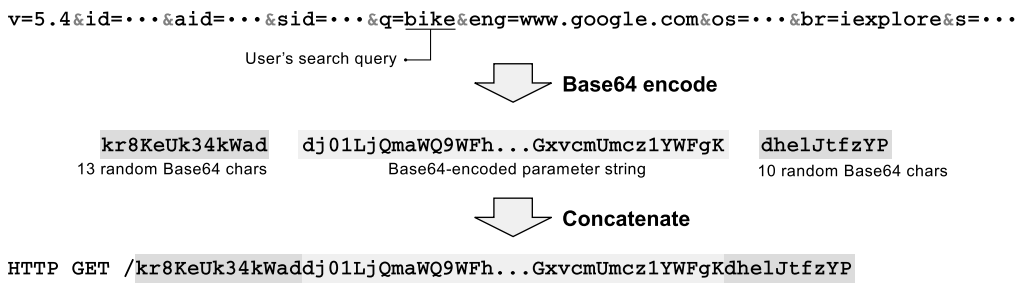


Figure 9.3: Encoding of a ZeroAccess search-hijacking module search request. When the user issues a search query, the module requests a list of URLs to which the user should be redirected. The user's query and other module parameters are combined using the standard URL parameter encoding scheme. The resulting string is then Base64-encoded and padded by prepending 13 and appending 10 apparently random Base64 encoding characters. The resulting string is then used to form the HTTP GET request to the SH-C&C server. (Values denoted "···" have been truncated for space in this example.)

```
v=5.4&id=%08x&aid=%u&sid=%u&q=%.*s&eng=%.*s&os=%s&br=%S&s=%u
```

The parameter string is then encoded using Base64 encoding and padded with 13 randomly-generated[5] characters at the front and 10 similarly-generated characters at the end. The length of the padding is such that a trivial decoding of the entire string does not reveal the contents of the message. The resulting string is then sent to the SH-C&C server in the HTTP GET request.

[5]The malware generates the padding characters using the Windows random number API.

IP Address	v=	*Pseudo-Domain*	*Purpose*
195.3.145.108	5.4	dclixvfpttrlcnindvrnyeic.com	Search request
	5.4	evtrdtikvzwpscvrxpr.com	
	5.3	atenrqqtfrzozqrqbdzwkxzyuc.com	
83.133.120.186	5.4	gozapinmagbclxbwin.com	Search request
	5.4	nbqkgysciuuhadgpjfquvpu.com	
	5.3	cjelaglawfoyidgyapv.com	
83.133.120.187	5.4	jpciukjdkqxgreoikpgya.com	Search request
	5.4	qhdsxosxtvmhurwezsipzq.com	
	5.3	omakfdwkhrpqudxvapy.com †	
217.23.3.225	5.4	hzhrjmeeczcgxodmqyz.com	Search request
	5.4	fnyxzjeqxzdpeocarhljdmyjk.com	
	5.3	sqdfmslznztfozshtidmigmsbh.com †	
217.23.3.242	5.4	vdlhxlmqhfafeovqohwrbaskrh.com	Search request
	5.4	nmfvaofnginwocnidecxnpcs.com	
	5.3	euuqddlxgrnxlrjjbhytukpz.com †	
188.40.114.195	2.1	qvhobsbzhzhdhenvzbs.com	Click confirmation
188.40.114.228 *	2.1	mbbcmyjwgypdcujuuvrlt.com	Click confirmation
	2.0	wuyigrpdappakoahb9.com	
217.23.9.247	6.1	vzsjfnjwchfqrvylhdhxa.com	Flash player identification
	5.8	vjlvchretllifcsgynuq.com	
83.133.124.191 *	5.6	chvhcncpqttfpcibtmetg.com	Flash player identification
178.239.55.170	1.2	jgvkfxhkhbbjoxggsve.com	Unknown / JavaScript injection
83.133.120.16	1.2	xlotxdxtorwfmvuzfuvtspel.com	Unknown / JavaScript injection
	1.1	mkvrpknidkurcrftiqsfjqdxbn.com	
83.133.124.191	—	ezcfogjitbqwnornezx.com	Fallback
	—	rwdtklvrqnffdqkyuugfklip.com	
	—	uinrpbrfrnqggtorjdpqg.com	
188.40.114.228	—	jzlevndwetzyfryruytkzkb.com	Fallback
	—	glzhbnbxqtjoasaeyftwdmhzjd.com	
	—	kttvkzpwufmrditdojlgytxyb.com	
46.249.59.47	—	loanxohaktcocrovagkaa.com	Fallback
	—	mxyawkwuwxdhuaidissclggy.com	
	—	erspiwscuqslhjflgbbgcfbc.com	
46.249.59.48	—	spujplpdupiwbghiedhqeja.com	Fallback
	—	xttfdqrsvlkvmtewgiqolttqi.com	
217.23.9.140	—	dxgplrlsljdjhqzqajkcau.com	Fallback

Table 9.5: Pseudo-domains and IP addresses extracted from the search-hijacking module via malware executions and reverse engineering. The v= column shows the value of the v argument used in requests. The *Purpose* column lists the class of commands sent to the SH-C&C server. Communication attempts to (and DNS requests for) *Fallback* IP addresses occur when the malware is unable to establish communication with a pseudo-domain selected for another function. When this occurs, the original SH-C&C message is sent to the fallback IP address. In this case, a pseudo-domain corresponding to the fallback address does *not* appear in the HOST field; instead, the original pseudo-domain appears. Pseudo-domains labeled with † were discovered via reverse engineering, but not verified in observations of network requests, presumably due to limited executions. IP addresses labeled with * reflect addresses that were unexpected given the associated pseudo-domains. This anomalous behavior occurred in a very small number of executions, perhaps due to some kind of bug, or related to the fallback domains. We inferred the IP addresses associated with predicted but not observed pseudo-domains via the de-obfuscation algorithm.

IP Address	*Location*
217.23.3.223	Netherlands
83.133.127.85	Germany

Table 9.6: IP addresses of the first hop servers in the ad click redirection chain for the search-hijacking module.

Primary Rendezvous

Hardcoded into each version of the ZeroAccess search-hijacking module is a list of `.com` domains of the form shown in Table 9.5. However, these domains are not resolved in the usual way using the Domain Name System. Instead, each domain encodes an IP address directly. To make the distinction clear, we call these *pseudo*-domain names. To obtain the IP address of a command-and-control server, the module decodes one of the pseudo-domain names to an IP address using the following algorithm given in which we extracted from the module binary.

```
from binascii import crc32
from struct import pack
from socket import inet_ntoa

def deobfuscate_domain(d):

    b0 = crc32(d[0:5],0x7E873D53) & 0xFF
    b1 = crc32(d[5:9],0x570848EB) & 0xFF
    b2 = crc32(d[9:12],0x768772F3) & 0xFF
    b3 = crc32(d[12:17],0x4775114F) & 0xFF

    ip_as_int = b0 + (b1 << 8)  \
                   + (b2 << 16) \
                   + (b3 << 24)
    packed_ip = pack('<I', ip_as_int)

    return inet_ntoa(packed_ip)
```

In addition to decoding to an IP address, the domain name may have been used for authentication as described in the next section.

Table 9.5 lists all pseudo-domain names and their associated IP addresses and domain names associated with the search-hijacking module that we observed. In addition, Table 9.6 lists the IP addresses of the first hop servers in the search-hijacking ad click redirection chain.

Authentication

Normally, an HTTP interaction progresses as follows: (1) the browser resolves the domain name in the URL (e.g., `www.google.com`) to an IP address; (2) the browser connects to the Web server

at the given address; (3) in its request, the browser sends a `Host` header specifying the domain name. Instead, the bot client skips the first step and from the pseudo-domain directly extracts the associated IP address encoded in the name. It then connects to the Web server and still includes a `Host` header with the pseudo-domain, even though it never resolved that name, and in fact could not since the name is unregistered in some cases.

During our early exploration of the botnet, we observed that manipulating or removing the `Host` header resulted in the SH-C&C protocol responding to messages with errors. After subsequent updates to the SH-C&C protocol, however, we were unable to reproduce this behavior.

This behavior, when active, could reflect usage of the domain name as way to authenticate legitimate bot clients to the SH-C&C server. No normal Web browser can reach the server via the domain name since it is not registered; and presumably no scanner trying to find Web servers will know which domain name to include in the `Host` header to look like a bot client.

Encryption

In response to a SH-C&C message, the server sends back an HTTP octet-stream of RC4 encrypted ciphertext. The response to search result C&C requests, once decrypted, provides a list of 0-or-more replacement ad URLs. These URLs are used in Step ❺ of the hijacking process.

The target for each ad URL is a first hop ad server (Table 9.6), which when visited will begin a 302-redirect chain. Along with this ad click URL is another URL to be used as a forged Referer field in the subsequent ad fetch.

Rate Limiting

During our interaction with the ZeroAccess SH-C&C we observed advertisement click rate limiting. When we performed frequent searches from a particular IP address, the SH-C&C initially returned a large (more than 5) number of ads per query. The more we interacted with the advertisements, though, the fewer ads were returned from subsequent servers. Specifically, we observed rate limiting across the following dimensions independently: source IP address, search term, and affiliate ID.

Rate limiting based on IP address or affiliate ID may imply the botnet attempting to limit the amount of fraud performed by a particular entity, in order to avoid detection. Rate limiting based on search term may reflect a limited supply of relevant ads for a particular term.

Secondary Rendezvous

Prior to establishing a connection to an IP addresses derived from an obfuscated SH-C&C domain, the ZeroAccess malware performs a DNS request (an `A-Record`) for the domain. This DNS request is generated by the ZeroAccess malware, and does not use any of the traditional Windows API's for resolving domains. The request always has DNS transaction ID `0x3333` and is always sent to Google's DNS server at `8.8.8.8`.

Through both static reverse engineering and live malware executions, we have been unable to ascertain the purpose for this DNS activity. When a new version of the search-hijacking module

is released, the domains associated with that module are generally not registered. Throughout the lifetime of the module various domains will become registered, sometimes by security researchers. However, the behavior of the ZeroAccess malware does not appear to be affected by the response to these DNS requests. The malware never uses the IP addresses returned by these queries, and its execution continues independent of the resolution status of the domain.

Other malware families have used similar functionality as a secondary rendezvous utilized to regain control of the botnet in the event of a takedown [87]. Although we do not believe the current ZeroAccess versions have such behavior, we are unable to definitively rule out such behavior.

Additional Functionality

In additional to the primary SH-C&C message that sends search terms and receives replacement ad URLs, there are three other distinct types of SH-C&C communication. The four SH-C&C messages have the same behavior with respect to how messages are formatted, obfuscated, and encrypted. These messages correspond with the *Purpose* categories in Table 9.5. Each of the four message types use the same primary rendezvous technique, although each pull from a distinct set of domain names.

The three additional SH-C&C messages have the following syntax:

```
v=1.2&id=%u&aid=%u&sid=%u&os=%s
v=6.1&id=%08x&aid=%u&sid=%u&os=%s&fp=%s&ad=%u
v=2.1&id=%08x&aid=%u&sid=%u&kw=%s&url=%s&ref=%s&os=%s
```

Click confirmation: After a user's click is hijacked, the malware sends a message of type `v=2.1` to a SH-C&C server, reporting the click URL as the URL parameter. In response to this message the SH-C&C server may direct the malware to perform additional clicks.
Flash player identification: Messages of type `v=6.1` report the user's Flash Player version to SH-C&C servers. The version is relayed via the `fp` parameter.
Unknown / JavaScript injection: The intended purpose of type `v=1.2` messages is unknown. In practice these messages occur far less frequently than the other types of communication. In response to this message from the malware, the SH-C&C will occasionally respond with ad network JavaScript, which we suspect is then injected into webpages viewed by the user.

Module History

Between May 2013 and November 2013 we have observed two distinct versions of the search-hijacking module (independent of changes to the included pseudo-domains). Initially (May and part of June), the `v` parameter observed in search request messages was `5.3`. During this time the response to search queries was a list of plaintext advertisements. In June, the value of the `v` parameter changed to `5.4`, and the response to search queries took on the encrypted form described above.

The `v` identifier for the other three categories of commands has also changed over time, as shown in Table 9.5.

We have also examined samples that had different pseudo-domain names hardcoded into them. Despite including different pseudo-domain names, the names generally decode to the same set of IP addresses.

Advertising Networks

Over time, from our observations and others [149], this module has been seen to defraud a large number of ad networks, including 7search, Affinity, Domain Development Corporation and Hoist Media. Some of these ad networks overlap with those seen defrauded by the click fraud module, but some we have only seen defrauded by the search-hijacking module.

9.6 Summary

In this chapter we have described two ZeroAccess modules, the *auto-clicking* module that performs traditional click fraud by simulating user clicks on advertisements, and a more recent *search-hijacking* module, which intercedes upon user clicks on Web search results, instead sending the user to an advertisement related to the search. In both cases, the botmaster stands to earn a commission from the click. We documented technical specifics for both forms in detail, including key infrastructure components (domain names and IP addresses corresponding to C&C servers) we have discovered via reverse-engineering of the modules and by observation of the malware's live execution.

Chapter 10

Characterizing Large-Scale Click Fraud in ZeroAccess

10.1 Introduction

Click fraud is a scam that hits a criminal sweet spot by both tapping into the vast wealth of online advertising and exploiting that ecosystem's complex structure to obfuscate the flow of money to its perpetrators.

In this chapter we illuminate these problems by characterizing the click fraud perpetrated by *ZeroAccess*. Active in a variety of forms since 2009 [56], ZeroAccess was one of the largest botnets in operation, commanding an estimated 1.9 million infected computers as of August 2013 [109]. More importantly, ZeroAccess was particularly known for monetization primarily via click fraud (with losses to advertisers estimated at $2.7 million per month [149]). However, while the technical aspects of ZeroAccess's design and operations (e.g., infection vector, peer-to-peer C&C protocol) are well documented [56, 109, 149, 150], the nature of its click fraud behavior and the attendant monetization has seen less study. In part, this is because such analyses require a range of disparate vantage points, including of the botnet itself, of infected hosts, and of impacted publishers.

By combining an array of data sources, including peer-to-peer measurements, C&C telemetry from botnet infiltration, and click information from one of the top ad networks, we have constructed a deep analysis to illuminate the rich, intertwined nature of modern click fraud and the advertising ecosystem it exploits. In particular, our work makes three contributions: First, we provide a detailed description of the click fraud component of ZeroAccess, the innovations it introduced in hijacking high-quality user search traffic, and the side effects by which we were able to track its activity. Second, we show how to match botnet membership data, network telemetry, and ad network click streams using a combination of timing information and reactions to external events (in this case the Microsoft-initiated takedown of ZeroAccess click fraud infrastructure and the botmaster's immediate responses). Finally, using this technique we identify with high confidence 54 individual "ad units" (here roughly corresponding to distinct traffic sellers) whose traffic volume (and hence revenue) was predominantly rooted in ZeroAccess. By anchoring our analysis in these ad units,

we roughly estimate that the botnet produced on the order of a million fraudulent clicks a day, plausibly inducing advertising losses on the order of $100,000 per day. However, the uncertainties involved in extrapolating to this global picture loom large enough that we must caution that this reflects a coarse-grained estimate, and accordingly we discuss the challenges involved in forensic accounting of click fraud payouts.

Taken together, this chapter illustrates the complex nature of the click fraud problem and highlights the need for much better mechanisms for correlating traffic and payment streams.

This chapter is based on work that appeared at the ACM Conference on Computer and Communications Security (CCS) [115].

10.2 Data Sources and Quality

Our study draws upon extensive, disparate sets of data. The individual datasets all suffer from limitations or skews of various forms. Constructing our large-scale picture in a sound fashion frequently requires cross-correlating different data sources in order to filter out spurious activity and bring out the underlying signals that reflect different facets of ZeroAccess's click fraud activity. In this section we sketch each of the data sources, our use of it, and its associated data-quality issues. A summary of all data used in this study is given in Table 10.1.

Click data. Via our partnership with one of the top ad networks, we acquired access to the "clicks" that the network logged from Nov 28–Dec 6 2013, spanning the Takedown event. The ad network views this data as highly sensitive from a business perspective and thus in our study we use the data at reduced fidelity and present certain facets of it only in relative terms rather than using absolute values.

In abstract terms, each "click" datum consists of a count of ad-following Web requests observed arriving at the ad network from a given source, during a given interval, and associated with a given ad unit. In the context of this work, an ad unit maps roughly to distinct traffic sellers. In particular, our click data aggregated individual clicks into tuples consisting of $\langle$ one-hour interval, /24 subnet, ad unit, count $\rangle$. We believe this data to be of high quality: none of our analyses or cross-checks raised questions regarding any potential inaccuracies or missing values.

Ad unit data. Our ad network partner also provided information for selected ad units in terms of the conversion percentage for their ads, and their mean and median smart-pricing discounts, with these latter being in normalized form so as not to reveal sensitive business information. The data included those ad units we identified as very likely tainted by ZeroAccess activity, as well as two randomly sampled populations of comparable size, and global baseline figures aggregated across all ad units.

This data covers the same time period as the click data discussed above. It allows us to explore the relative effects of ZeroAccess's activity compared to regular ad unit costs and conversion efficacy.

ZeroAccess DNS telemetry. As discussed previously, vestigial code in Serpent's modules leads it to issue DNS requests to a number of different domains based on its current activity. Each different Serpent function has a different set of domains associated with these lookups; as far as our

Dataset	Granularity	Quantity
ZA DNS telemetry, Dec 1–Dec 4		
Timestamp	millisecond	16,208,758
Domain	Full query	12
IP	/24	336,609
Supernode data, Dec 1–Dec 6		
Timestamp	millisecond	260,811,204
IP	/32	1,137,118
IP	/24	637,736
Bot type	32/64 bit OS	2
ZA module distribution, Jun 18–Apr 25		
Timestamp	second	51
Module ID	64-bit ID	51
Module MD5	Full MD5 sum	51
Milker data, Sept 10–Dec 5		
Ad replacements	Full URL	1,766
Redirects	Full URL	10,796
Click data, Nov 28–Dec 6*		
Timestamp	hour buckets	Over 10TB of raw ad server logs
IP address	/24	
Clicks	hourly sum	
Ad unit	Anon ID	
Ad unit data, Nov 28–Dec 6*		
Timestamp	hour buckets	Around 2TB of raw ad server logs
IP address	/24	
Clicks	hourly sum	
Ad unit	Anon ID	

Table 10.1: Summary of datasets used in our study. *Precise quantities for click and ad unit data intentionally omitted due to business sensitivities. We only use this data in aggregate.

extensive analysis could tell, the malware chooses randomly among the set for a given function. The malware does not process any replies it receives for associated DNS requests, and thus does not even require that the domains exist.

Indeed, during our study most of the domains did not exist. Beginning on Nov 28 2013 we registered all such ZeroAccess domains not already registered (6 out of 12), and immediately began receiving queries for them. The queries all came from Google's public DNS server, `8.8.8.8`, and thus nominally did not identify the associated ZeroAccess system. However, Google includes support for an EDNS0 option [32] that identifies the subnet originally associated with requests that resolvers such as theirs issue. Thus, our data from this source has the form of tuples consisting of ⟨ timestamp, domain, /24 subnet ⟩, where the timestamp has high precision (sub-second) as recorded at our DNS server.

The domains we registered included 3 of the 5 associated with Serpent modules reporting that users had searched, and the sole domain associated with Serpent reporting that a user had clicked on a substituted search ad. (The other domains related to functionality not relevant for our click fraud study.)

At first blush this data held promise for illuminating the fine-grained activity of (nearly) each Serpent infectee. However, extensive analysis of the data revealed that lookups did not have a one-for-one correspondence to individual Serpent actions. The data also included significant activity clearly associated with timers, but not so sharply timer-driven that we could readily distinguish it from legitimate activity. However, the data *does* provide us with a virtually complete list of IP addresses associated with ZeroAccess's Serpent module, which allowed us to cross-check against ZeroAccess supernode data to assess its completeness.

Supernode data. From a partner we acquired a list of nodes discovered by crawling the ZeroAccess peer-to-peer network from Dec 1–6 2013. Each entry consists of ⟨ timestamp, address, ZA-network ⟩, where timestamp is high precision, address is the full /32 IP address, and ZA-network reflects on which of the two ("32-bit" and "64-bit") P2P networks the crawl found the node.

Note that these nodes should reflect a *superset* of Serpent nodes, since not all ZeroAccess infectees ran Serpent. Thus, from a click fraud perspective this data is potentially more complete than that derived from the DNS data described above. However, by cross-checking the /24s seen in the DNS data with the equivalent /24s seen in this data, we found that the supernode data included only 200,708 of the 336,609 Serpent /24s. This discrepancy is explained by the design of the P2P crawl resulting in a incomplete view of the P2P network. Thus we conclude that this data reflects only about 60% of the entire ZeroAccess population. This shortfall becomes crucial in our subsequent analysis as we aim to determine which ad units present in the ad network's click data clearly had significant ZeroAccess-generated clicks in their traffic.

ZA module distribution information. Beginning on Dec 4 2013, we ran ZeroAccess infectees in 56 long-running, contained VM environments to allow them to participate in the P2P network and thus receive module updates. Whenever one of them received a new module, we detected the event in real-time and captured a copy of the module. This data allows us to track the evolution of the botnet's functionality.

In particular, this data source allows us to track the botnet's partial recovery post-Takedown

(when the auto-clicking modules were updated), which we use in our subsequent analysis as one of the signals for identifying ad units whose traffic has significant taint from ZeroAccess activity. It also allows us to study the blending of Serpent traffic with auto-clicking.

Milker data. Drawing upon extensive reverse engineering, we developed an emulator for the ZA-C&C protocol used by Serpent to request ads to substitute into those present in a user's search results. The emulator enabled us to "milk" ad replacements out of the C&C server by repeatedly requesting ads from it, though the C&C server appeared to "dry up" in its ability to provide new replacements after repeated queries. (The rate at which this drought occurred varied.)

Each C&C reply provided a URL to click on (along with a matching Referer). We followed the URL using a fully functional headless Web browser displaying the `curl` User Agent (such a User Agent prevents advertisers from being charged from our seeming clicks), which in general would continue for each click until it ends at an advertiser's landing page. In total, we captured the redirection chains for 1,766 such clicks for a small sampling of search terms we selected from trending shopping queries. 367 of the clicks transited our partner ad network.

Based on our DNS telemetry correlated with our click data, and supported in part by our small scale milking experiment, we believe that the global impact of ZA was likely an order of magnitude larger than seen by our partner ad network.

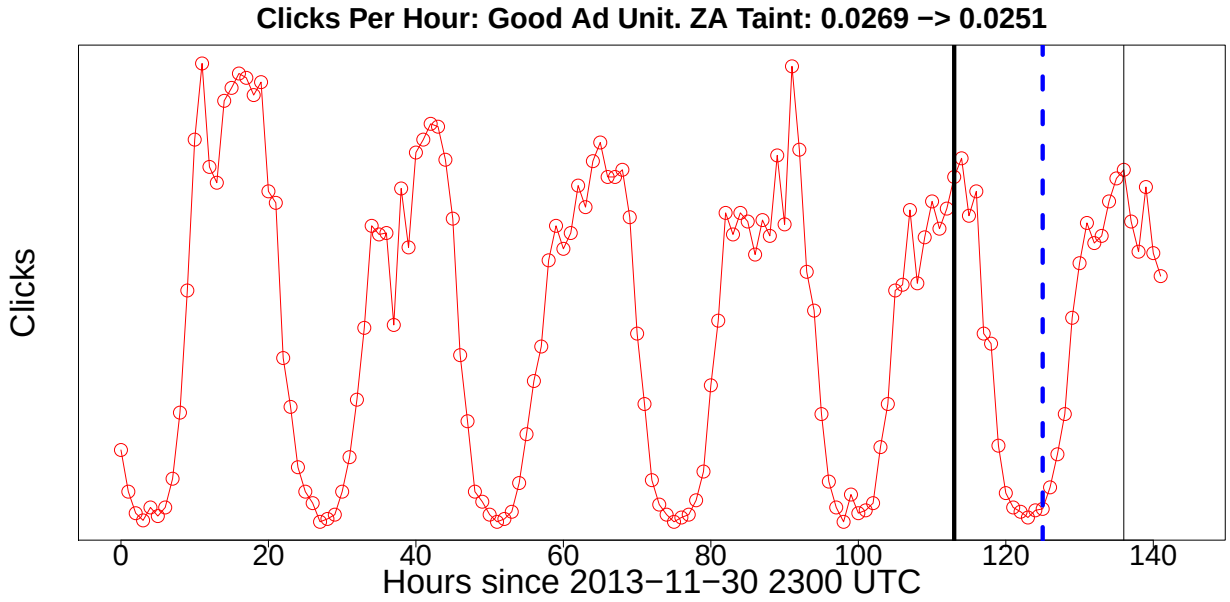
10.3 Analyzing Fraud

To build up our overall picture of ZeroAccess's large-scale click fraud activity, we start with the data most central to assessing ZeroAccess's impact, namely the Click data provided by our ad network partner. We then draw upon our other data sources to identify activity (primarily ad units) associated with ZeroAccess clicks. We employ two main approaches—analyzing the behavior of sources to the Takedown event, and looking at source "demographics" in terms of which subnets contribute clicks—and then cross-correlate these two to develop our overall picture.

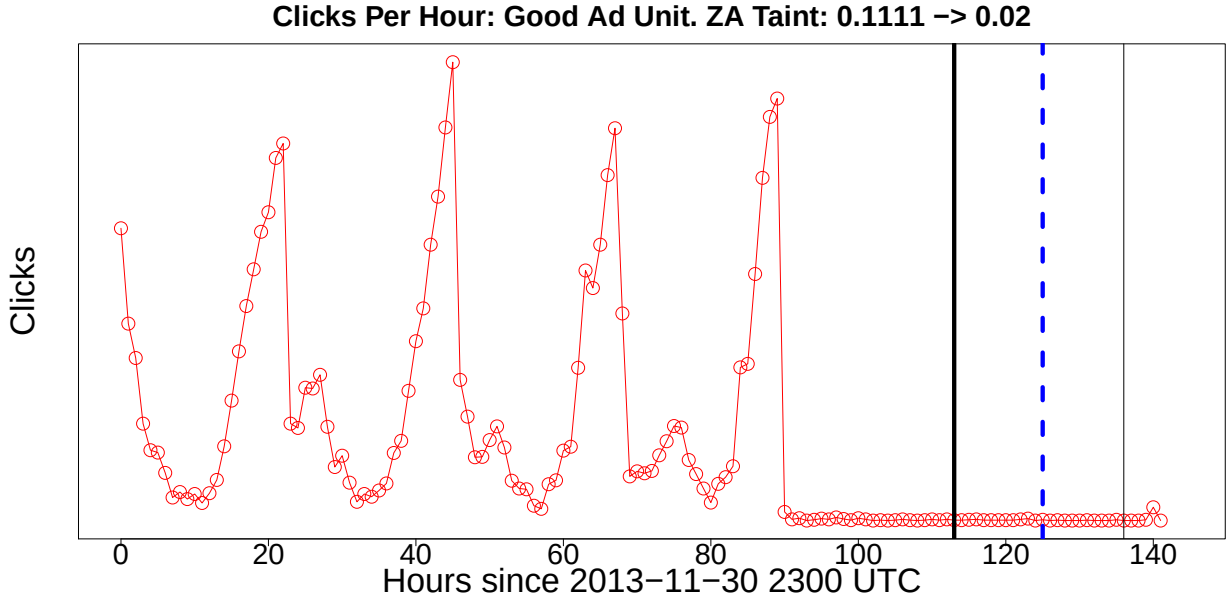
Takedown Dynamics

The Dec 5 2013 Takedown event abruptly severed C&C for ZeroAccess's click fraud activity, which in principle should manifest as a striking change in the activity of any source fueled by ZeroAccess clicks. We then face the basic question of how to reliably detect this presumably sharp signal without inadvertently treating benign variations in traffic rates as stemming from ZeroAccess.

Figure 10.1 shows the relative clicks per hour for four exemplary ad units, with the leftmost vertical line marking Takedown. Here we have partitioned the examples into "good" ad units that do not primarily receive ZeroAccess traffic and "dirty" ones that do (using a methodology we will describe shortly). Normal click behavior prior to Takedown exhibits the expected diurnal pattern. For good ad units, this pattern continues (Figure 10.1a), while for dirty ones, precisely at Takedown their clicks cease or dramatically decrease (Figures 10.1c and 10.1d).

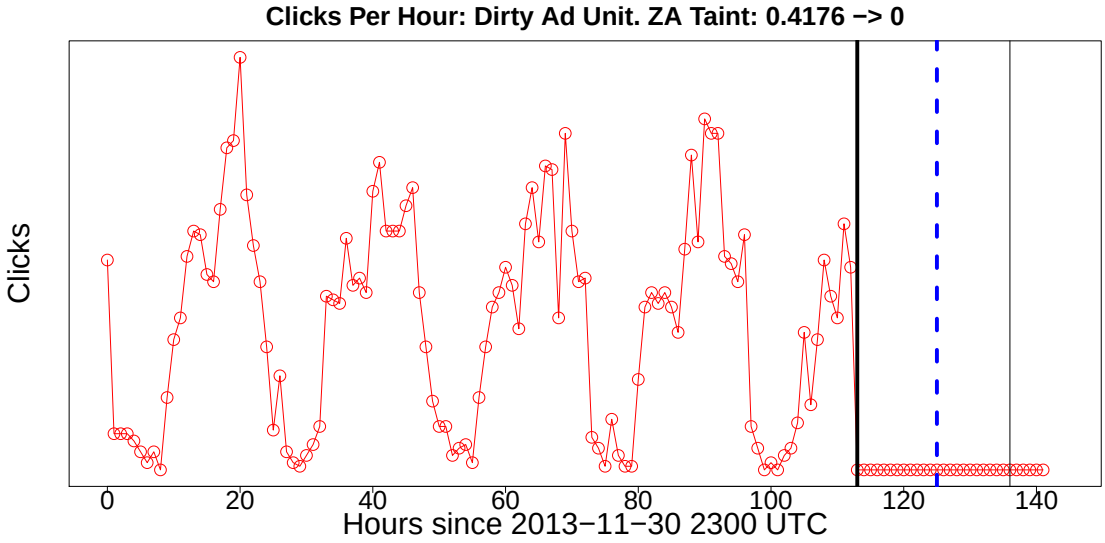


(a)

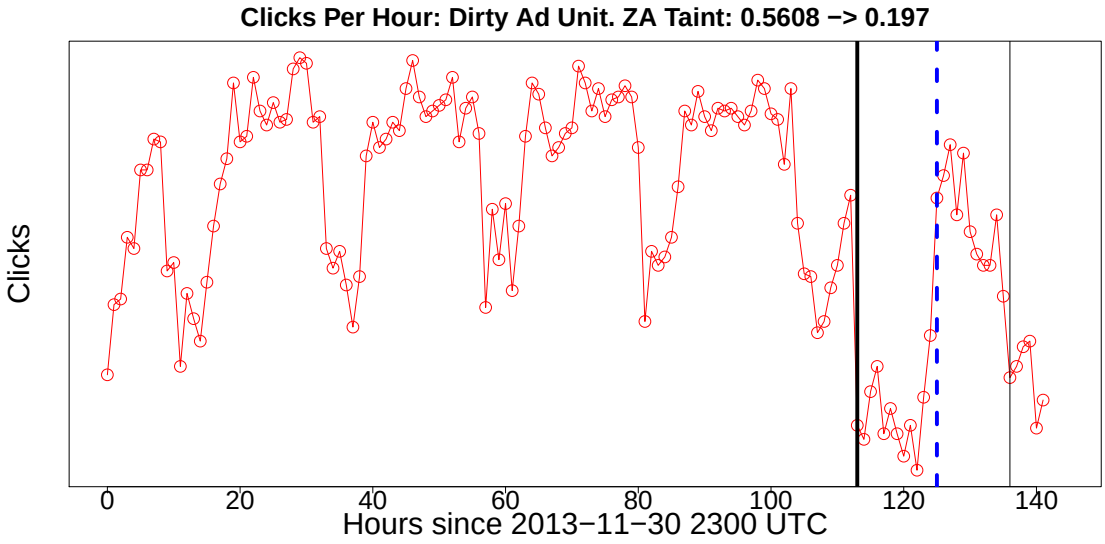


(b)

Figure 10.1: (Figure continued on next page.)



(c)



(d)

Figure 10.1: Clicks per hour prior to and after Takedown for 4 exemplary ad units. The thick vertical line marks Takedown (8AM PST, Dec 5 2013). The dotted line to its right indicates the release by the ZeroAccess botmaster of a new auto-clicking module to counteract the Takedown. The thinner line to its right indicates the "WHITE FLAG" module release (per Section 8.1). The plot titles also give the "ZA taint" (Section 10.3) pre- and post-Takedown (per Section 10.3). 10.1a shows a typical large ad unit that does not see a significant drop in click traffic post-Takedown. The ad unit in 10.1b exhibits a large drop in traffic that occurred *prior* to Takedown, highlighting the surprising benign dyanmics manifest in the data. 10.1c shows an ad unit whose traffic almost entirely consisted of fraudulent clicks. The ad unit in 10.1d clearly had a substantial proportion of ZeroAccess traffic, but mixed in with legitimate or non-ZeroAccess clicks.

However, we cannot simply attribute ZA-dirtiness to *any* source that precipitously fell at the time of Takedown, because such behavior also manifests for benign sources (per Figure 10.1b). Such behavior can be attributed to advertising budget depletion or the end of specific advertising campaigns. This behavior necessitates the need for incorporating multiple signals to soundly identify dirty sources.

One such signal concerns the effects of a secondary Takedown event. The dotted line in Figure 10.1 corresponds to the release of a new auto-clicking module as a response to the attempted takedown. This module contained new C&C IP addresses, and resulted in auto-clicking click fraud resuming for a subset of the botnet. We can see that some sources exhibit a spike in activity coincident with this module's appearance (Fig. 10.1d) while others do not (Fig. 10.1c).

Armed with these signals, we attempted to robustly identify dirty sources based on statistical testing. We undertook numerous evaluations looking for robust indications of behavioral shifts. For example, we compared the volume of each source's click activity as seen during the hour of the Takedown (denoted H-hour) versus during the previous hour ($H-1$). Using the null hypothesis that the relationship between these counts remained unaffected by the Takedown, we applied Fisher's Exact Test to assess the consistency of the shift between those hours as seen on a non-Takedown day versus that seen on Takedown day.

The test identified a large number of sources with statistically significant deviations in the shift for those hours, even for quite low p values (e.g., 0.001).[1] Manual inspection of the most extreme examples confirmed that many appeared to clearly reflect instances of ZeroAccess-affected behavior.

However, when we then tested one non-Takedown day against *another* non-Takedown day, we likewise found many statistically significant deviations, which clearly had nothing to do with the Takedown and thus presumably nothing to do with ZeroAccess activity. The clear conclusion (backed up by manual assessments of exemplars) is that the null hypothesis often fails to hold due to frequent *non-stationarity* in the data. That is, a given click source's patterns from one day to the next can exhibit striking variations; two separate days are not well-modeled as independent samples from the same underlying population. (Figure 10.1b shows such an instance.)

This lack of stationarity significantly complicates our analysis, and means that statistical testing can only serve as a guide to help direct manual analysis due to the risk of false positives. (In addition, the non-stationarity serves as a caution for applying any sort of training-based machine learning to the problem of identifying fraudulent ad click sources.)

Subnet overlap

Conceptually separate from the Takedown dynamics, we can seek to identify dirty ad units by assessing each source's degree of "ZA taint" (i.e., proportion of individual sources potentially associated with ZeroAccess activity). This taint can then provide us additional context with which to interpret a given source's Takedown dynamics.

[1] We used a one-sided test since we only had interest in a shift towards an abnormally low H-hour level.

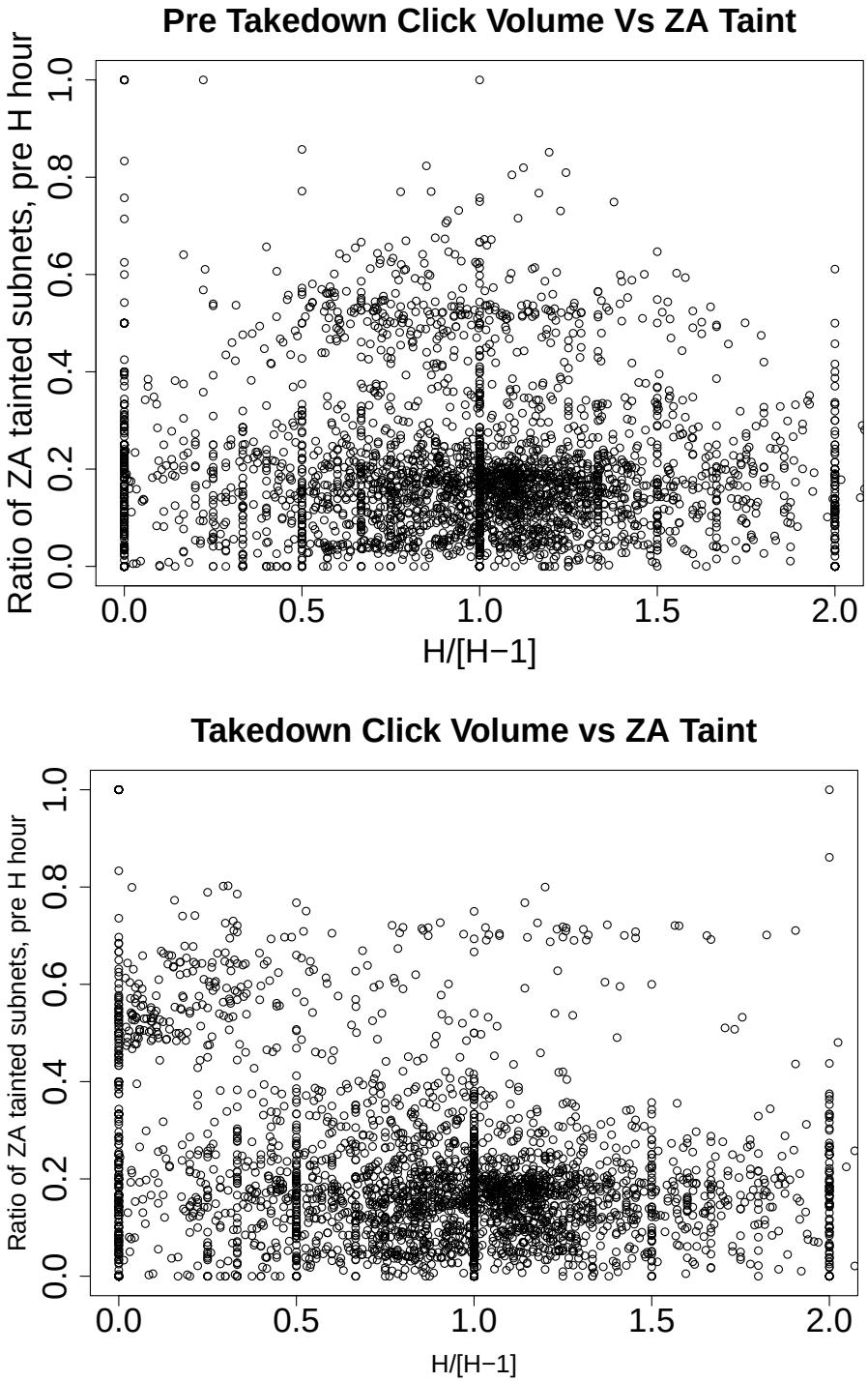


Figure 10.2: Comparison of ad-unit click volume before and after the Takedown hour (H-hour) to the amount ZeroAccess taint. The left shows this comparison on the day prior to Takedown, and the right across Takedown. A large population of ad units has both higher-than-average ZeroAccess taint and also shows a strong shift towards much fewer clicks on Takedown. The harmonics at $x = 0.5$, $x = 1.0$, $x = 1.5$, etc., arise from ad units with very low click volumes.

ZA taint. For each ad unit, we consider its full set of clicks. Due to restrictions of the dataset, we identify clicks based on one-hour granularity and /24 subnet of the source of each click. For each hour, we compute the proportion of subnets appearing in the ad unit's traffic that also appeared during that hour in our Supernode data. We then term the mean value of that proportion as the ad unit's "ZA taint". (We limit this computation to hours up to but not including H-hour, so as to not skew the taint by the Takedown dynamics.)

Limitations of the supernode comparison. The Supernode data suffers from incompleteness. By the nature of the ZeroAccess P2P network, no single node has a complete vantage point of the entire network, and thus the matching for our taint computation may incur significant false negatives. To gauge the impact of these false negatives, we compared the ZeroAccess supernode data with our DNS telemetry. Our DNS telemetry gives us complete /24 subnet views of the Serpent portion of the ZeroAccess botnet, but no vantage of the auto-clicking portion. Still, any Serpent subnet missing from the Supernode data likely reflects incompleteness in the latter. If we look for DNS telemetry subnets to show up within 1 hour in the Supernode data, then we find about an 80% match. If we look for matches within 1 minute, this drops to 60%. Infectees repeatedly show up in the Supernode data with such frequency that this latter comparison may in fact provide a better estimate than the former (which allows for a degree of IP address churn to introduce false positives).

A further limitation of our data is the reduction of address information to /24 subnets, which results in us tainting all clicks from a given subnet as bad. Such false positives will result in some benign ad units having increased ZeroAccess taint.

Combining Signals

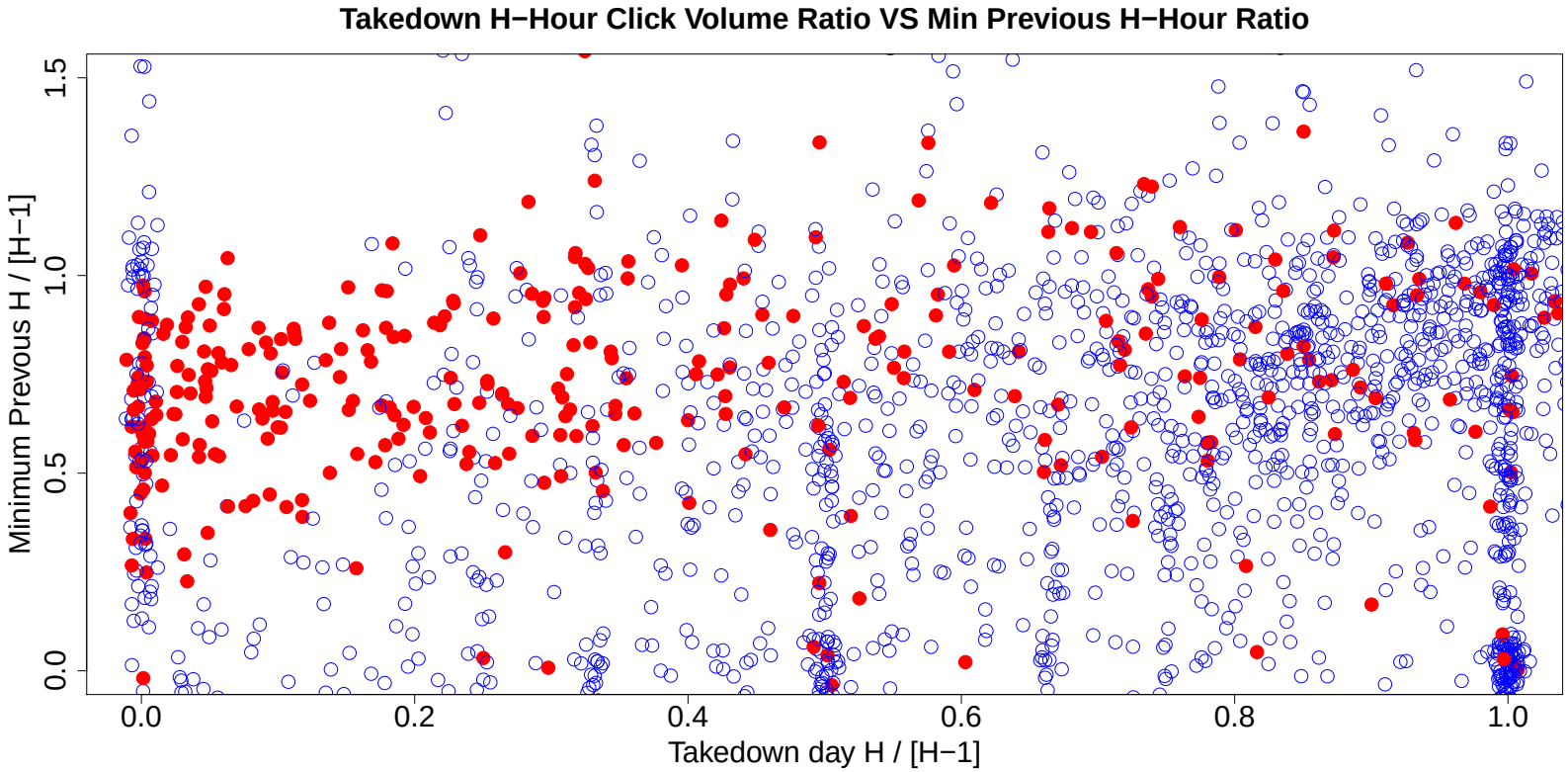


Figure 10.3: Comparison of the stability of traffic drops across H-hour per ad unit. We plot the Takedown H-hour ratio against the minimum H-hour ratio seen on any other day. We denote ad units with ZA taint ≥ 0.4 using solid red circles, and others with hollow blue circles. Note that we have added small amounts of Gaussian jitter to prevent coincident points from completely overlapping.

Given the limitations of the Supernode data, we cannot use the presence of these IPs as a sole indicator. Instead, we look for ad units that exhibit both a high fraction of ZeroAccess taint as well as uncharacteristic behavior at the various Takedown-related events.

To combine notions of taint and Takedown response, we calculate the ratio of the amount of click traffic at H-hour to the previous hour, $\frac{H}{H-1}$. Ratios closer to 0 denote sharp drops in traffic. Dirty ad units might not exhibit a ratio of exactly 0 because of traffic from other sources, or click-fraud events already "enqueued" from prior to the Takedown.

Figure 10.2 shows a comparison of the traffic-drop ratio to ZeroAccess taint for each ad unit. We do this for H-hour on both the day before Takedown (left) and the day of Takedown (right). The first plot shows a population of ad units with higher-than-average ZeroAccess taint centered around a drop ratio of 1 (no major change across H-hour) on the day prior to Takedown. The second plot shows a dramatic shift to a greatly reduced volume of traffic at H for that same population of ad units with high ZA taint, with a large number of them approaching zero traffic.

Figure 10.3 explores the stability of traffic of each ad unit across H-hour over time. We plot the Takedown traffic ratio across H-hour against the minimum H-hour ratio for all previous days. Solid red circles denote ad units with ≥ 0.4 ZA taint. A large population of ad units with high ZeroAccess taint exhibits atypically low H-hour ratios (below 0.5) compared to their previous minimum (above 0.5). We deem these ad units as likely ZeroAccess dirty.

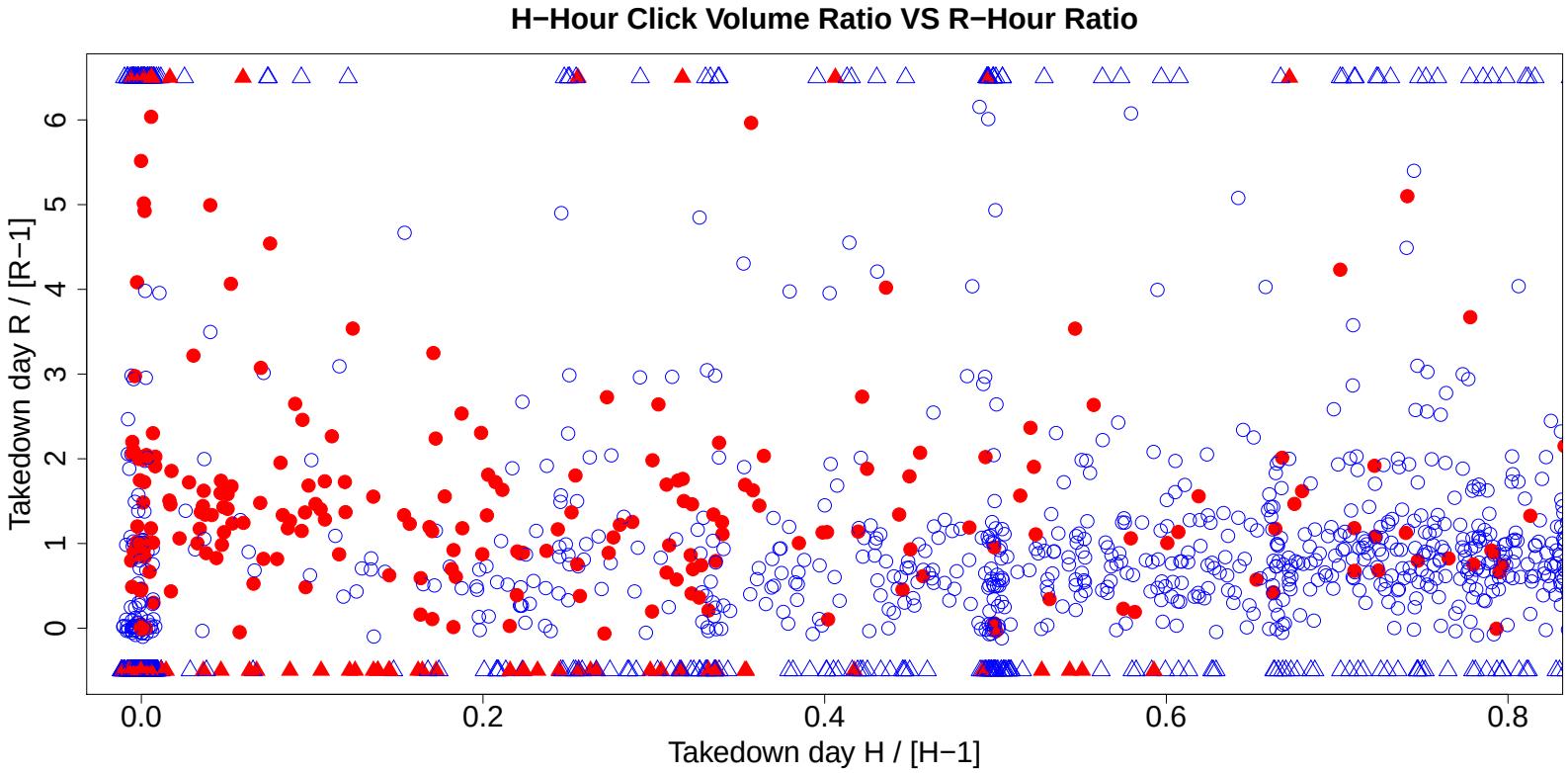


Figure 10.4: Ad unit response to Takedown (X axis, in terms of H-hour ratio) versus to subsequent dissemination of new auto-clicking module (Y axis, in terms of R-hour ratio). Solid red circles indicate ad units with ZA taint ≥ 0.4. Triangles below the Y origin reflect ad units that had no traffic during either $R-1$ or R hours. Triangles above $Y = 6$ had traffic during R-hour but not during $R-1$. Note that we have added small amounts of Gaussian jitter to prevent coincident points from completely overlapping.

Secondary takedown information. Figure 10.4 shows the behavior of each ad unit across another Takedown event, the distribution of the auto-clicking module shortly after Takedown, which we denote as *R*-hour. Not all ZeroAccess infectees ran the new module, but those that did would result in a dramatic rise in click activity at *R*-hour compared to just before *R*-hour; a change in the opposite direction as what we would observe at *H*-hour. The figure shows the traffic drop ratio across *R*-hour plotted against the drop across *H*-hour, again using solid red circles to denote ad units with $\geq$ 0.4 ZA taint. Clearly, many ZeroAccess-tainted ad units showed sharp reactions to both Takedown and the advent of the new module.

The distribution of an updated auto-clicking module at *R*-hour immediately restarted ZeroAccess's click fraud. We would expect the resumption of click fraud to result in dirty ad units having low ZeroAccess taint in the time between *H*-hour and *R*-hour (since no C&C servers exist to drive click fraud), followed by a sudden increase in taint at *R*-hour.

Figure 10.5 compares taint as seen during these two regions of time (*H*-hour to *R*-hour, and after *R*-hour). The left plot reflects a (presumably typical) non-Takedown day, while the right plot shows Takedown day. The shift between the two is clear: a large number of ad units that had significant taint prior to Takedown (solid red) show a jump in pre-*R* vs. post-*R* taint, reflecting the activation of a significant number of ZeroAccess subnets within their traffic.

Finalizing determination of dirty ad units. Using these signals we generated a large set of potentially dirty ad units, about 2,000 in number. We then manually inspected the activity of each and produced a list of 54 we deem with high confidence as significantly "dirty" with ZeroAccess traffic.

10.4 Assessing ZA-Dirty Ad Units

Having identified these dirty ad units, we now employ them as the basis for evaluating the strategy used by the more sophisticated Serpent module to circumvent smart-pricing. We also leverage this conservative set of ad units to estimate the total amount of fraud perpetrated by ZeroAccess, an undertaking that strongly emphasizes the need for better attribution in the ad ecosystem.

Conversions and Smart-pricing

We next use our data to explore potential reasons for why ZeroAccess may have used both auto-clicking and Serpent styles of click fraud. Recall from Section 8 that ad networks use automated approaches to either detect (and block) specific click fraud attacks, or mitigate the impact of click fraud (e.g., through smart-pricing) when ad networks can only estimate the amount of click fraud in the aggregate. Clearly Serpent, which confuses a real user into clicking, is harder to detect (and block) than auto-clicking.

We first examine whether we see evidence that Serpent indeed increases the chance of users converting and, if that is the case, whether the blending of Serpent and auto-clicking avoids the smart-pricing mitigation.

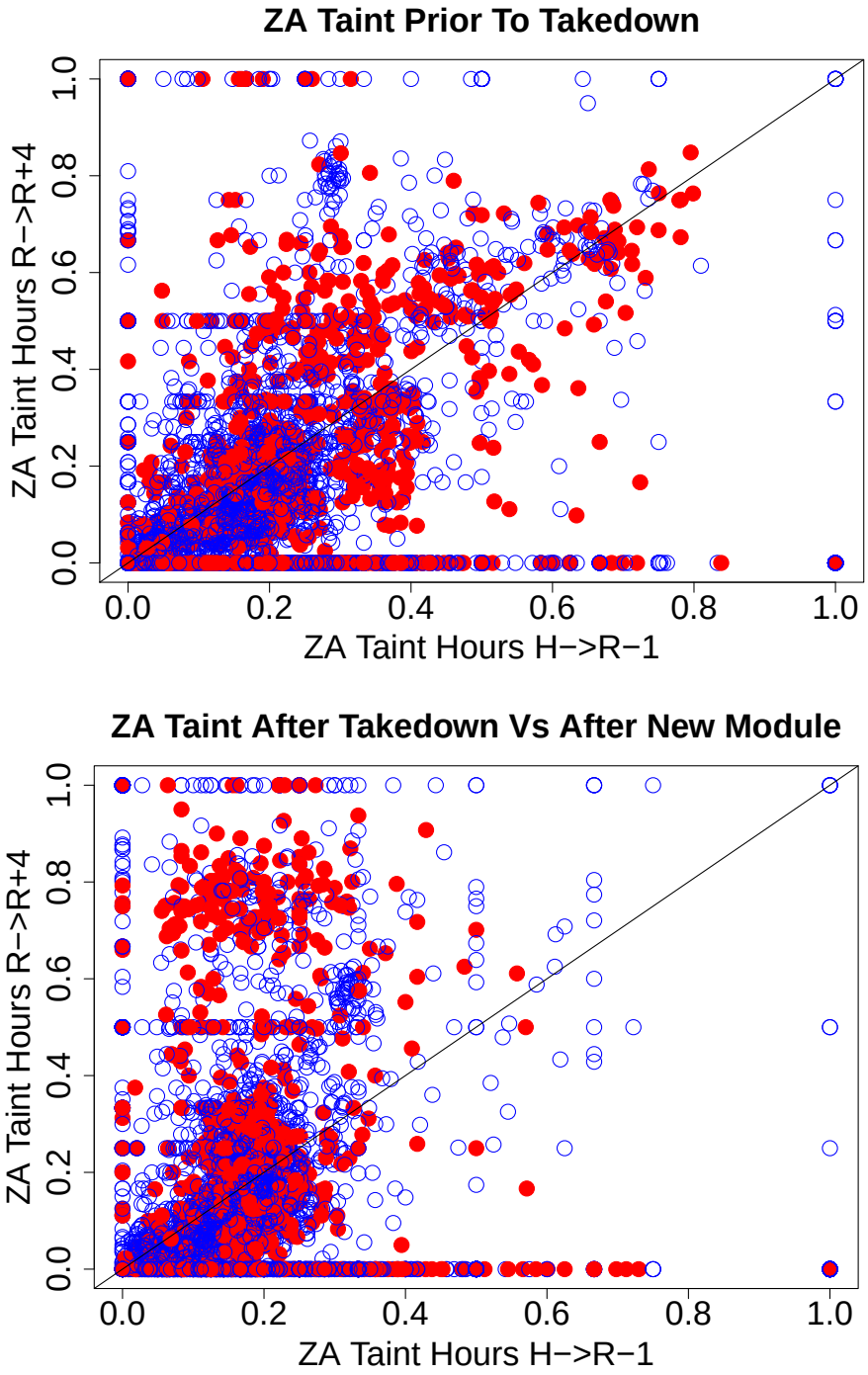


Figure 10.5: Comparison of ZA taint before and after R-hour, per ad unit. The X axis gives taint from between the Takedown hour and just before the hour corresponding advent of the new auto-clicking module post-Takedown; the Y axis as computed over the 4 hours starting with R-hour. The left plot shows the day prior to Takedown, the right plot the day of Takedown. Solid red circles reflect ad units that had a drop in H-hour ratio across the day before and after Takedown ≥ 0.5 (this corresponds to the population shift in Figure 10.2).

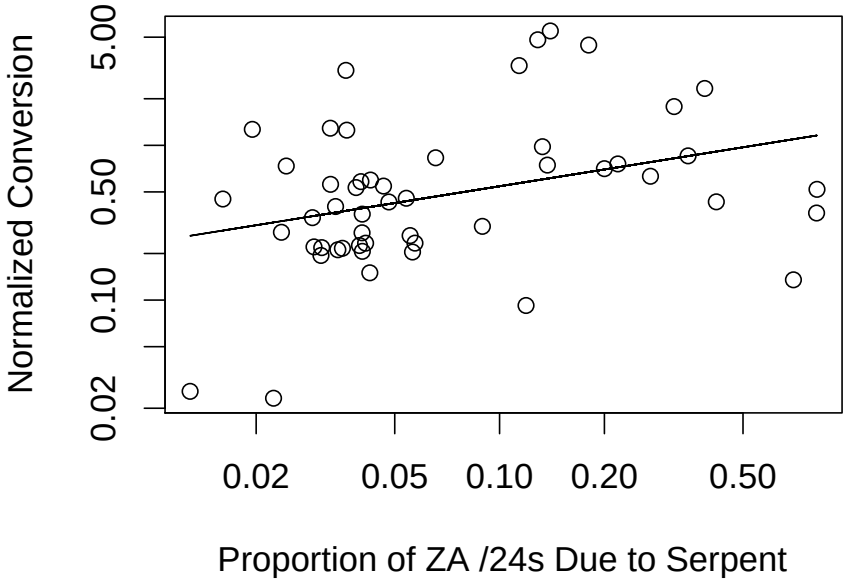


Figure 10.6: Relationship between the proportion of ZeroAccess traffic that includes Serpent and the conversion rate for the dirty ad units. The X axis gives the proportion of an ad unit's ZeroAccess subnets that also appeared in the DNS telemetry data, and thus include Serpent activity. The Y axis reflects the normalized conversion rate across the set (1.0 = the mean rate of the set). We log-scale both axes, discarding 3 ad units that had either no Serpent subnets or never converted. The line shows a least-squares fit to the log-transformed data, corresponding to a power-law relationship with an exponent of about 0.36.

Figure 10.6 shows a clear but modest correlation between ZeroAccess's use of Serpent (via ad units with increased Serpent taint) and growing conversion, though with significant variation across ad units. Keep in mind that other than Serpent, all of the ad unit's conversions will be due to legitimate user activity and not ZeroAccess. This result shows that the use of Serpent does add conversions of fraudulent click activity.

Next, to understand if the combined blending of Serpent and auto-clicking avoided the smart-pricing mitigations, we compare our sample of 54 dirty ad units to a random sample of ad units of similar size. We find that compared to a random sample, those flagged as ZeroAccess dirty had conversions rates of up to three times less. As a result, these dirty ad units were indeed subject to the smart-pricing discount. Thus, it appears that if the intent of the Serpent module leveraging real users was to avoid smart-pricing, then it failed to do so since the users forced to go to the advertiser pages did not convert enough. If the intent behind leveraging real users, however, was simply to

avoid detection while accepting the smart-pricing discounted income, then it was successful.

Estimates and Challenges

Formulating a sound estimate of the global impact of ZeroAccess proves very difficult, not only due to limitations in our data sets, but more fundamentally because of the sharply limited visibility that the ad network ecosystem provides. Using some plausible assumptions, we can derive estimates of ZeroAccess's impact perhaps within an order of magnitude, and note that these paint roughly a similar picture to that developed in the work of others [149]. Clearly, however, the important question of just which parties make how much profit due to this illicit activity poses thorny methodological challenges.

Limitations within our ad network. Although our ad network partner provided access to the complete click payout information for the 54 dirty ad units we identified, those ad units make up only a subset of the likely total ZeroAccess traffic abusing their network. Methodologically, we require an ad unit to exhibit a baseline level of traffic before we can label it as fraudulent (via manual analysis). Ad units, however, exhibit a very long tail in their level of activity. The vast majority provided (individually) so little traffic that although they exhibited numerous ZeroAccess characteristics, such as high ZeroAccess taint, for any given such ad unit we could not rule out these characteristics occurring simply by chance, and thus could not definitively label them as bad. In aggregate, however, such ad units potentially account for a large volume of traffic, much more than those of the 54 definitively dirty ad units that we identified. How to soundly attribute a portion of that traffic as fraudulent remains an open question.

Limitations across ad networks. A further complication arises in identifying the breakdown of fraud between our ad network partner versus other networks targeted by ZeroAccess. Our "milking" data provides a qualitative impression of what this balance might look like, finding that 367 of the 1,766 click chains transited our ad network partner. Superficially this suggests that in total, ZeroAccess click activity might be about five times what our ad network partner sees. However, along with the limited scale of the milking data, other basic factors, such as how presented ads (and thus click chains) are influenced by IP address or search term, could result in major shifts in this ratio. In addition, the milking data only reflects one instance of Serpent activity, and we do not know if its distribution of ad networks accords with that of the auto-clicking module.

Limitations in DNS data. While our DNS telemetry gives us visibility into Serpent ad clicks, the data suffers from significant noise. Moreover, we do not know how to extrapolate from pure Serpent activity to the behavior of the auto-clicking module. When merging the DNS data with other sources, we also encounter issues with both the granularity of the IP address information and IP address churn, making attribution within our partner's ad network problematic.

Assumptions and estimates. Given those major caveats, we now formulate some rough estimates to get a sense of the overall scale of activity.

The 54 dirty ad units we identified generated over 100K clicks per day. It appears certain that aggregate ZeroAccess activity far exceeds this lower bound, which we view as very solid (we have high confidence those ad units all reflect ZeroAccess activity) but also very conservative. Building on this foundation, given the large volume of suspicious-but-not-definitive ad units we

investigated—both the "long tail" with dubious but sparse activity, and those that appear to carry ZeroAccess traffic blended with non-ZeroAccess traffic—we would argue that likely the total activity within our ad network partner was at least a factor of $2x$ larger.

If we then take the $5x$ factor seen in our limited click chain milking to represent the volume of traffic seen in other ad networks, we have a total extrapolation of $10x$ to our original conservative estimate, yielding 1M fraudulent clicks per day. This range is consistent with the order of magnitude of fraudulent clicks estimated from our DNS data.

According to our ad partner, the cost-per-click of this traffic after smart-pricing normalization would be between 10–30 cents. Taking the low end of these ranges, we can construct an estimate of the total click fraud impact of ZeroAccess as on the order of $100,000 per day, which aligns with the estimate from Sophos of ZeroAccess generating up to $2.7 million in click fraud per month [149].

We highlight, however, that enough uncertainties exist that our figure can only reflect a "best guess" estimate. We also note that given the available data sources, we lack the ability to gauge what fraction of this money makes it into the hands of the botnet owners after the long chain of (sub)syndicates take their cut.

Discussion

Subsyndication is a problem. Subsyndication is the key business model enabling large-scale click fraud. Even a single (trusted) publisher that is allowed to subsyndicate opens the flood gates for click fraud traffic (blended with other legitimate traffic) entering the ad network unwittingly through that publisher. Our Serpent milker encountered syndication chains as long as 13 domains deep. It is no surprise therefore that we were able to definitively conclude significant ZA dirtiness for only 54 ad units. There were many more where our analysis was inconclusive due to high levels of blending that diluted the otherwise sharp signal presented by the Takedown.

One might argue that *anonymous* subsyndication is the real problem, and if publishers could be forced to identify themselves, the search engine would be able to better police the ad network. While this is true, we feel it is impractical. The level of blending we see suggests that many publishers are complicit in click-fraud, especially since everyone along the syndication chain benefits financially from it. Even if the search engine could recursively force (sub)syndicates to set a policy that requires identifying their traffic sources—a hard problem in itself—there would be little reason for the (sub)syndicates to enforce it.

Low signal to noise ratio. Our analysis would not have been possible absent the strong signal injected into the ZeroAccess botnet by the Takedown. Even then we had to combine three large datasets from three different vantage points. Such strong signals are too few and far between to be an effective means of policing ad networks. The capability to inject more frequent (but perhaps less intense) disruptions into botnets could create a strong temporal signal that could serve as a basis for ongoing click fraud policing.

10.5 Summary

In this chapter we undertook a detailed examination of the activity of one of the largest click fraud botnets in operation, ZeroAccess. Through reverse-engineering and controlled execution, we mapped out ZeroAccess's click fraud component, including the innovations it introduced in hijacking high-quality user search traffic. In the process we discovered side channels in the form of vestigial DNS lookups that enabled us to track its large-scale activity. We then combined this perspective with partial "supernode" data capturing the botnet's peer-to-peer C&C activity in order to match up botnet activity data with ad-unit click stream data provided by our major ad network partner.

By leveraging the striking shifts induced in click activity by Microsoft's takedown of ZeroAccess in Dec 2013, along with the ZeroAccess botmaster's subsequent response, we combined these diverse data sources to identify with high confidence 54 individual ad units whose traffic volume (and hence revenue) primarily derived from ZeroAccess. Our ensuing analysis of these ad units revealed that while the latest generation of click fraud botnets have circumvented many detection approaches, they are still mitigated (for now) by the smart-pricing mechanism. Extrapolating from the known-bad ad units, we constructed an estimate that ZeroAccess likely generated on the order of a million fraudulent clicks per day across all ad networks, with the overall ecosystem revenue diverted by the botnet's activity roughly on the order of $100,000 per day.

Finally, we note that the complexity of the online advertising ecosystem is such that no one party comes remotely close to having comprehensive visibility into who gets paid what for any given click. The hodgepodge of data sources required for our analysis starkly illustrates both the tangled nature of the click fraud problem space and the pressing need for much better mechanisms for correlating traffic and payment streams.

Chapter 11

Ad Injection at Scale: Assessing Deceptive Advertisement Modifications

11.1 Introduction and Background

Subsequent to the rise and fall of large-scale botnet-driven fraud (Chapters 7-10) a new development in the landscape of advertising abuse emerged: *ad injection*. In this scam, ads are injected into a user's browsing experience by either a network attacker or software running on the user's system. Since these advertisements are appearing in front of real users, the impressions and clicks they generate are *real*, making the identification and remediation of the problem difficult. [1] As browser extensions have gained wide-spread use, an ecosystem of deceptive extensions that monetize installations via these injected advertisements arose. Such extensions operate on an affiliate model, similar to ad syndicators, allowing extension authors to directly monetize installations by entering into relationships with ad injectors. These injectors provide libraries to extension authors which are then executed when users visit websites while the extension is loaded.

Ad injection differs from traditional advertising abuse in the open, legal nature in which these extension authors and syndicators operate [86]. Broadly, the collection of extensions and software that operate in this legal gray area are termed "unwanted software," denoting the likelihood that users did not willingly or knowingly consent to their installation, and do not desire their services. The most notable example, Superfish, gained notoriety when it was discovered pre-installed on Dell PCs in 2015 [86] [2]. In addition to injecting ads, these extensions can reduce user safety by introducing vulnerabilities into the browsing process [86].

Money enters the ad injection ecosystem from advertisers. Advertisers' willingness, perhaps unwittingly, to pay for traffic generated by these injectors drives the proliferation of ad injection.

[1] The concept of co-opting real user behavior is similar to the ZeroAccess malware [115,118]. ZeroAccess hijacked non-ad user clicks, rewriting the destination, and ultimately turning those clicks into click fraud. The user may have then engaged with the advertiser, stemming from hijacking confusion. Ad injection, however, inserts additional ads into a user's experience, causing real impressions and potentially further engagement from an *interested* user.

[2] This work predates the discovery of Superfish's installation on Dell hardware, and the subsequent media attention [143].

The revenue from these advertisers sustains not only the ad injectors but also an entanglement of intermediaries that connect advertisers to injectors.

Building on work with Google which identified and executed these deceptive extensions in order to extract the revenue chains injected into various websites, this chapter focuses on exploring the ecosystem of intermediaries and advertisers that support the ecosystem through analysis of injected click (revenue) chains.

Our collaborators built systems that identified and analyzed the advertising injection behavior from 50,870 browser extensions and 34,407 binary samples. For this analysis, we focused on 114,999 click chains injected by 398 browser extensions into three websites (termed *properties*) that are well known to experience ad injection: `Amazon.com`, `Google.com`, and `Walmart.com`. Chains were extracted by visiting these various websites, executing relevant representative search queries, and then extracting the injected ad URLs. This chapter focuses on unraveling the complex web of advertisers and intermediaries supporting the ecosystem.

In this context, an intermediary in the ad injection ecosystem is a business entity that in some way connects advertisers to injectors. These entities can take on a variety of forms including traffic resellers, traffic exchanges, and online product aggregators.

We find that although a handful of online retailers comprise the majority of advertisers encountered via ad injection, a large tail of less prolific advertisers are the subject for most of the ads. We also identify a "choke point" of three ad networks and 25 affiliate programs that were responsible for the majority of ad injections.

This chapter starts with an overview of the ecosystem of advertisers and intermediaries involved in ad injection. Following this overview, we closely examine the three largest ad injectors to identify the major advertisers and intermediaries that these ad injectors depend on. We then characterize the relationships among these three groups to understand the extent to which advertisers and intermediaries are aware of ad injection and what potential bottlenecks exist in this ecosystem.

This chapter is based on a portion of work that appeared in the IEEE Symposium on Security and Privacy (S&P) [135].

11.2 Impacted Properties

Using network traffic gathered from our dynamic execution of browser extensions [135], we reconstructed and analyzed 114,999 ad revenue chains in order to understand the relationships between ad injectors, intermediaries, and advertisers in this ecosystem. Based on the construction of the dynamic execution engine, injections were only identified for a handful of top impacted properties.

Table 11.1 provides a breakdown of which properties are impacted in the ad revenue chains we collected. 62,237 of our chains were injected into amazon.com, 37,718 were injected into google.com, and 15,044 were injected into walmart.com. A majority of extensions injected ads into all three properties, with the average number of hops in the chain varying based on the impacted properties.

Impacted Property	Triggered Queries	Click Chains	Ad Injection Libraries
amazon.com	90	62,237	27
google.com	96	37,718	13
walmart.com	71	15,044	25

Table 11.1: Breakdown of each impacted property and its contribution to the dataset.

Injected Ad Domain	Click Samples	Avg Hops	Advertising Domains
superfish.com	63,891	3.6	891
dealply.com	20,209	9.2	526
datafastguru.info	3,899	4.9	407
display-trk.com	3,353	12.7	196
tfxiq.com	3,091	1.8	58
pangora.com	1,814	2.9	204
xingcloud.com	1,116	1.0	4
shoppingate.info	994	4.9	232
linkfeed.org	822	6.3	267
bestyoutubedownloader.com	688	1.0	1
Other	15,122	7.48	1,497

Table 11.2: List of the top 10 ad injectors we observe in our extension execution.

11.3 Understanding Injectors in Revenue Chains

In revenue chains, ad injectors are identified as the first hop domains, beginning with the impacted property. We group together domains that share affiliation, as well as resolving shorteners and content distribution networks (CDNs) to the next hop in the chain.

Table 11.2 is a breakdown of the top 10 injectors appearing in ad revenue chains. Ad chain volume is heavily skewed towards a handful of injectors, with the top two injectors accounting for 73% of all injections. As is the case with client detection, Superfish is the dominate injector in our revenue chains, accounting for 56% of all chains (originating from 60% of extensions). The next most prevalent injector is the Dealply group. Dealply is a grouping of domains that share affiliation (IP addresses, ad feeds) with the Dealply program. Dealply accounts for 18% of revenue chains (29% of extensions). Datafastguru is the next largest contributor to revenue chains, accounting for only 3.4% of chains (25% of extensions). As is the case with client detection, there is a long tail of injectors contributing to the complete ecosystem.

The diversity of advertisers fueling each ad injector also varies greatly between injectors. Some injectors (xingcloud.com, bestyoutubedownloader.com) have very few advertising landing

domains, likely indicating that the injector is the same party as the advertiser. Meanwhile other injectors are supported by huge number of advertisers. Amongst top injectors, Superfish is supported by the fewest advertisers, relative to click volume. Superfish injections average 1.39 advertisers per 100 clicks. Dealply is supported by nearly double that rate, with an average of 2.60 advertisers per 100 chains. Strikingly, Datafastguru has a much larger relative base of support, with a rate 4x that of Dealply, at 10.44 advertisers per 100 clicks.

The diversity of advertisers for each of these top programs is explored in more depth in Section 11.4.

Revenue chain length varies dramatically between ad injectors, with some injectors having few to no intermediaries on average (tfxiq.com, xingcloud.com) and others having a large number (display-trk.com). Shorter revenue chains denote fewer intermediaries (for a given click), and perhaps simpler business relationships stemming from fewer syndication agreements. Shorter chains may also denote more advertiser visibility into the origin of clicks. Superfish revenue chains have a relatively small number of average hops, 3.61. Dealply chains, comparatively, have average length 2.5x that of Superfish, at 9.15 hops.

11.4 Advertisers and Intermediates For Top Injectors

By examining the intermediaries and advertisers associated with the top three ad injectors in greater depth, we shed light on the relationships and affinities between intermediaries, advertisers, and injectors. These three injectors, Superfish.com, Dealply.com, and Datafastguru.info account for 77% of all injections originating from 81% of extensions.

Advertiser Composition

While the ecosystem of ad revenue chains are extremely complex and intertwined, there are a handful of advertisers that are responsible for most of the revenue chains. The top five advertisers observed in traffic from each injected ad revenue chain are given in Table 11.3.

Online retailers (Sears.com, Walmart.com, Target.com, Wayfair.com, and Overstock.com) makeup virtually all of the top advertisers supporting Superfish, Dealply, and Datafastguru. Exceptions are Kobobooks.com, an eBook company, and Bizrate.com, a product aggregator.

Sears.com and Walmart.com account for almost 30% of the advertising landing pages reached via Superfish injections, the dominate injector. Although these retailers each contribute to a sizable portion of Superfish injections, the total advertiser population is long-tailed, with over 800 domains totaling 59% of total chains.

Dealply and Datafastguru advertisers have a similar long tail, but with the relative makeup of the top advertisers differing. The long tail of advertisers for these programs also contributes the majority of total volume. The dominate advertiser supporting Datafastguru is ebay. While ebay is present in the Dealply ecosystem, it is not a dominate source of advertising. The difference arises from the much larger number of Datafastguru advertisers (relatively).

Ad Injector	Advertiser (.com)	% of Inject.	Avg Hops	Common Intermediaries and Co-Occurrences
superfish	sears	18%	3.25	DT,CI 100%
	walmart	11%	2.83	PG 43%; DT 40%
	kobobooks	6%	2.86	PG 92%
	target	4%	5.11	CI 100%; BR,CI 53%
	wayfair	2%	4.27	BR 85%
	Other	59%	3.82	PG 37%; DT 30%; BR 28%
dealply	target	12%	10.50	CI 100%; CI,SF 51%
	wayfair	5%	9.39	BR 88%; SF 50%
	walmart	5%	10.44	BR 100%; BR,SF%
	overstock	4%	10.87	ME,BR 98%; ME,BR,SF 61%
	sears	3%	10.53	CI 100%; CI,SF 47%
	Other	71%	8.66	BR 59%; SF 38%; PG 21%
datafast	ebay	10%	2.23	DT 99%
	target	10%	5.76	CI 100%; BR,CI 51%
	bizrate	5%	1.15	None
	wayfair	4%	4.24	BR 63%
	sears	4%	5.37	CI 87%
	Other	68%	5.39	BR 49%; AD 19%

DealTime	DT	ChannelIntelligence	CI
PG Partners	PG	BizRate	BR
SuperFish	SF	Mercent	ME
Adnxs	AD		

Table 11.3: Top five advertisers for each of the top three injection programs. "Other" consists of all chains not belonging to the top five advertisers. For each advertiser, interesting common and co-occurring intermediaries are given.

Ad Injector	Intermediary	% of Ads	Hop After Injector	Hop Before Advertiser
superfish.com	dealtime.com	40%	26%	0%
	pricegrabber.com	34%	22%	9%
	channelintelligence	27%	0%	27%
	bizrate.com	23%	23%	1%
	searchmarketing.com	11%	0%	9%
	Other	18%	30%	54%
dealply.com	bizrate.com	67%	57%	2%
	superfish.com	43%	0%	9%
	channelintelligence	21%	0%	18%
	amung.com	19%	4%	1%
	clk-analytics.com	17%	0%	0%
	Other	78%	39%	70%
datafast.com	bizrate.com	43%	41%	1%
	frontdb.com	28%	0%	10%
	pronto.com	21%	15%	0%
	channelintelligence	19%	0%	17%
	adnxs.com	16%	0%	6%
	Other	67%	44%	66%

Table 11.4: Top five intermediaries for each of the top three injection programs. Injections is the percent of that program's chains for which a given intermediary was present.

Although the set of advertisers is long-tailed, the top 20 advertisers represent a significant bottleneck, comprising 50% of advertising clicks.

Intermediary Overview

Similar to advertisers, intermediaries in ad revenue chains are also long-tailed. Table 11.4 gives a breakdown of the top five most prevalent intermediaries for each of the top three programs.

Dealtime.com and Pricegrabber.com, both online product advertising aggregators, are the two most prevalent intermediaries in the Superfish ecosystem. While both exist in Dealply and Datafast-guru chains, they are not as prevalent. The two most common intermediaries across all three injectors are Channelintelligance.com (a Google property that links advertisers to ad networks) and Bizrate.

The Datafastguru ecosystem also introduces new intermediaries not seen in the other injectors, such as pronto.com, a price comparison service, and Adnxs.com, a property of AppNexus, which specializes in real-time ad exchanges.

Notably, Superfish has a presence in the Dealply ecosystem. In these chains Superfish appears to serve the role of an intermediary instead of an injector. In its capacity as an intermediary, Superfish plays a major role in driving traffic to all of the top Dealply advertisers.

Despite the large tail, only three intermediaries account for 57% of all first hops, representing a serious bottleneck in relationships between injectors and advertisers.

Chain Length

Table 11.3 contains the average number of hops for each advertiser, broken down by program. Sears.com and Walmart.com have a similar number of average hops (3.25 and 2.83, respective), both lower than the Superfish average of 3.60. These short chains denote a small set of intermediaries that link these advertisers to Superfish.

Dealply advertisers have distinctly more hops and intermediaries for the average chain. advertisers such as Walmart.com that had relatively short chains under Superfish (2.83 hops on average) have much longer chains (10.44 on average) under Dealply. This difference could denote Dealply has a more layered ecosystem with more syndication and traffic reselling than the other programs.

Intermediary Relationships With Injectors And Advertisers

Table 11.4 shows the likelihood of the top five intermediaries being either the first hop after an injector or the last hop before an advertiser. Such adjacency in the ad revenue chain denotes traffic exchanged directly between parties, if not a direct business relationship. Each of these affinities is given as the total percent of chains for a given injector (not a given intermediary). For short chains, a single hop may be both the first and last hop.

Across the Superfish ecosystem several interesting affinities exist. For chains that traverse Dealtime, almost all occur as the first hop, but less than half occur as the last hop. Pricegrabber shows similar behavior. Channelintelligence shows very different affinity, as it never appears directly following Superfish, and almost exclusively appears as the last hop prior to an advertiser. Bizrate has inverse affinity from Channelintelligence, with almost all appearances occurring adjacent to Superfish, and almost never adjacent to the advertiser.

None of the top intermediaries, besides Bizrate, show a clear likelihood for appearing adjacent to Dealply in ad chains. This denotes a long-tail set of intermediaries linked directly to Dealply, with this tail comprising more of the first hop volume than in the case of Superfish. Similar properties hold for Datafastguru. This denotes a large set of business relationships with Dealply.

Intermediary Co-Occurrence

Just as some advertisers show affinity to specific sets of intermediaries, some intermediaries show affinity for each other. Table 11.3 enumerates interesting co-occurrences of intermediaries per advertiser and program.

wayfair.com is reachable only via bizrate.com, and target.com is reachable via dealtime only when in conjunction with channelintelligence. sears is reachable only via channelintelligence, and kobobooks only via pricegrabber.

These commonly co-occurring intermediaries may represent limited advertiser relationships with ad networks supporting injection, also a potential bottleneck in injection revenue.

11.5 Summary

Ad injection is a lucrative and stealthy abuse scam that blurs the line between malware and software. By leveraging real user interaction and behaviors to generate advertising impressions and clicks, extension vendors and ad networks are able to monetize unsuspecting users' browsing.

In this chapter we perform a systematic analysis of 114,999 ad chains collected from 398 browser extensions, examining the ecosystem of advertisers and intermediaries that support this landscape. From this analysis we identify a structural "choke point" of three ad networks and 25 affiliate programs which are responsible for a majority of all ad injections, and can be leveraged for defense.

Chapter 12

Advertising Abuse Conclusion

In Part II we uncovered the scale, structure, and nature of advertising abuse across a variety of monetization strategies and attacks. Our multifaceted approach leveraged multiple methods, data sources, C&C infiltration, and industry data to describe attacks that by their very nature are difficult to identify. Our work identified fundamental structural weak-points leverageable for defense, resulting in the dismantling of botnets, cleaning up of ad networks, and protection of users.

We began in Chapter 7 [105] with an external, malware-driven execution study of click fraud malware and the ad networks that enable them. This work uncovered the breadth and scale of the ad abuse ecosystem, but its external-only visibility was a fundamental limitation that necessitated further study.

Next, in Chapters 8 and 9 [105, 118] we performed an in-depth analysis of ZeroAccess, a botnet and malware distribution platform that harmed millions of users and was responsible for millions of dollars in advertising fraud per month. Beginning with an overview of the malware and its evolution, we explored the botnet's structure, technical capabilities, and inner workings. This work helped facilitate a Microsoft-led takedown of the botnet [90, 104], resulting in the demise of the botnet and the cleanup of millions of infected PCs.

Chapter 10 [115] combined an array of external data sources, including P2P measurements and C&C telemetry, with an internal ad network perspective, to identify the scale and scope of ZeroAccess fraud. This multifaceted approach identified fraudulent business relationships within the ad network, and was used as a focal point for remediation.

The demise of ZeroAccess signaled the end of the large-scale, botnet-driven click fraud era. Subsequently new forms of advertising abuse, such as ad injection, began to emerge. Chapter 11 [135] examines the complex ecosystem of ad networks, syndicators, and intermediates that fueled the injection ecosystem. Our work identified and leveraged injector and ad network choke points within the ecosystem, enabling remediation for millions of users.

Chapter 13

Conclusion and Future Directions

The impact of global Internet security problems such as censorship and cybercrime continued to increase during the production of this dissertation. As systems and data also expand in complexity, the need for empirical grounding continues to be critical. This work presents methods and systems that not only enable the systematic understanding of these threats, but also open the door to new avenues of research. This section frames the contributions of this dissertation and how they inform future research directions.

Longitudinal and Continuous Censorship Measurement With the development of Augur and Iris, we now have the ability to perform *continuous* censorship measurement around the globe. This capability opens the door to new avenues of research: we can study the trends of censorship within a single country as well as across groups of countries. Using multiple vantage points within countries, we can begin to understand how censorship is deployed and the heterogeneity of deployment, at scale. Using this work we can identify the onset of new censorship and the dynamics of blocking behavior around key political or social events. The ability to perform controlled repeated measurements also allows us to understand the efficacy of various blocking techniques and evasions, and how censors respond to these evasions. Understanding these facets of censor behavior and technology will allow computer scientists to develop more effective, well-grounded defenses and evasions, while also enabling social scientists and policy makers to study the interactions between users and censors at a scale not previously possible.

Combating Cybercrime with Information Sharing Cybercrime remains an ongoing challenge for advertising networks. Systematic efforts across the community, including our work understanding and remediating ZeroAccess, had measurable impact. Subsequently, criminals shifting their techniques from bot- and malware-driven operations to unwanted software [135] and "in-house" (i.e., criminals emulating users themselves at scale) operations [145].

With this continued prevalence of cybercrime attacks, companies struggle to identify and remediate attacks at the speed and volume they occur. The ability to quickly and reliably share threat information (e.g., IP addresses, malware samples, URLs, tactics) between targets is emerging as a

popular proactive defense, and, though of unproven utility, has become current industry best practice. This trend is supported by recent legislation (the Cybersecurity Information Sharing Act), aimed at improving information sharing within the United States.

Within industry, this information sharing is commonly known as *threat intelligence*, an emerging market valued at over three billion US dollars. But despite significant industry capital and supporting legislation, the scope, scale, quality, and efficacy of threat intelligence remains unstudied. In order to answer these questions we need sound, well-designed evaluation metrics and criteria that will allow us to develop effective deployment strategies.

The emergence of this field coupled with the methods and systems of this dissertation present a unique opportunity to develop evaluation metrics, collection and curation methodologies, and effective sharing strategies—ultimately with the aim of changing how we defend systems by enabling effective multi-vantage data-driven defenses that do not exist today.

Exploring Advanced Persistent Threats Anecdotal industry and initial academic findings suggest state-sponsored advanced persistent threats (APTs) target a variety of organizations ranging from human rights activists to governments. These long-term, covert, highly-targeted threats use sophisticated malware, processes, and reconnaissance to manually compromise networks. The continued growth of these state-sponsored attacks highlights the need for empirical grounding and sound, rigorous understanding of these threats and how best to defend against them.

While this dissertation focused on criminals, many of the methods and systems are directly applicable to nation-state actors. This natural extension is enabling ongoing work examining APT target selection, recidivism, and ultimately a comprehensive understanding of how to reason about and defend against nation-state adversaries.

Bibliography

[1] M. Abu Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A Multifaceted Approach to Understanding the Botnet Phenomenon. In *ACM SIGCOMM*, 2006.

[2] G. Aceto, A. Botta, A. Pescapè, N. Feamster, M. F. Awan, T. Ahmad, and S. Qaisar. Monitoring Internet Censorship with UBICA. In *International Workshop on Traffic Monitoring and Analysis (TMA)*, 2015.

[3] Alexa Top Sites. `http://www.alexa.com/topsites`.

[4] C. Anderson. Dimming the Internet: Detecting throttling as a mechanism of censorship in Iran. *arXiv preprint arXiv:1306.4361*, 2013.

[5] C. Anderson, P. Winter, and Roya. Global Network Interference Detection Over the RIPE Atlas Network. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2014.

[6] G. Angioni. PokerStars, Full Tilt, and 888Poker Blacklisted in Latvia. `https://www.pokernews.com/news/2014/08/pokerstars-full-tilt-and-888poker-blacklisted-in-latvia-18971.htm`, August 2014.

[7] Anonymous. GreatFire.org. `https://en.greatfire.org/`.

[8] Anonymous. The Collateral Damage of Internet Censorship by DNS Injection. *ACM SIGCOMM Computer Communication Review (CCR)*, 2012.

[9] 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.

[10] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster. Building a Dynamic Reputation System for DNS. In *USENIX Security Symposium*, 2010.

[11] S. Aryan, H. Aryan, and J. A. Halderman. Internet Censorship in Iran: A First Look. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2013.

[12] M. Bailey and C. Labovitz. Censorship and Co-option of the Internet Infrastructure. Technical Report CSE-TR-572-11, University of Michigan, Ann Arbor, MI, USA, July 2011.

[13] BBC's website is being blocked across China. http://www.bbc.com/news/world-asia-china-29628356, October 2014.

[14] The Belmont Report - Ethical Principles and Guidelines for the Protection of Human Subjects of Research. http://ohsr.od.nih.gov/guidelines/belmont.html.

[15] S. Bodmer and M. Vandegrift. Looking Back at Murofet, a ZeuSbot Variants Active History. http://blog.damballa.com/?p=1008, Nov. 2010.

[16] D. Bolton. Putlocker Blocked in the UK by Internet Service Providers after High Court Order. https://goo.gl/s8Hb43, May 2016.

[17] G. Bonfa. Step-by-Step Reverse Engineering Malware: ZeroAccess / Max++ / Smiscer Crimeware Rootkit. http://resources.infosecinstitute.com/step-by-step-tutorial-on-reverse-engineering-malware-the-zeroaccessmaxsmiscer-crimeware-rootkit, November 2010.

[18] S. Bortzmeyer. Hijacking through Routing in Turkey. https://ripe68.ripe.net/presentations/158-bortzmeyer-google-dns-turkey.pdf.

[19] G. Buehrer, J. W. Stokes, and K. Chellapilla. A Large-scale Study of Automated Web Search Traffic. In *Workshop on Adversarial Information Retrieval on the Web*, 2008.

[20] J. Caballero, C. Grier, C. Kreibich, and V. Paxson. Measuring Pay-per-Install: The Commoditization of Malware Distribution. In *USENIX Security Symposium (USENIX)*, 2011.

[21] J. Caballero, P. Poosankam, C. Kreibich, and D. Song. Dispatcher: Enabling Active Botnet Infiltration using. Automatic Protocol Reverse-Engineering. In *ACM Conference on Computer and Communications Security (CCS)*, 2009.

[22] CAIDA. Archipelago (Ark) Measurement Infrastructure. http://www.caida.org/projects/ark/.

[23] A. Chaabane, T. Chen, M. Cunche, E. D. Cristofaro, A. Friedman, and M. A. Kaafar. Censorship in the Wild: Analyzing Internet Filtering in Syria. In *ACM Internet Measurement Conference (IMC)*, 2014.

[24] W. Chen, Y. Huang, B. F. Ribeiro, K. Suh, H. Zhang, E. de Souza e Silva, J. Kurose, and D. Towsley. Exploiting the IPID Field to Infer Network Path and End-System Characteristics. In *Workshop on Passive and Active Network Measurement (PAM)*, 2005.

[25] K. Chiang and L. Lloyd. A Case Study of the Rustock Rootkit and Spam Bot. In *USENIX Workshop on Hot Topics in Understanding Botnets (HotBots)*, 2007.

[26] C. Y. Cho, J. Caballero, C. Grier, V. Paxson, and D. Song. Insights from the Inside: A View of Botnet Management from Infiltration. In *USENIX Conference on Large-scale Exploits and Emergent Threats (LEET)*, 2010.

[27] D. Cicalese, D. Z. Joumblatt, D. Rossi, M. O. Buob, J. Aug, and T. Friedman. Latency-Based Anycast Geolocation: Algorithms, Software, and Data Sets. *IEEE Journal on Selected Areas in Communications*, June 2016.

[28] Cisco OpenDNS. `https://www.opendns.com/`.

[29] Citizen Lab. Block Test List. `https://github.com/citizenlab/test-lists`.

[30] Citizen Lab. `https://citizenlab.org`.

[31] R. Clayton, S. J. Murdoch, and R. N. M. Watson. Ignoring the Great Firewall of China. In *Privacy Enhancing Technologies Symposium (PETS)*, 2006.

[32] C. Contavalli, W. van der Gaast, D. C. Lawrence, and W. Kumari. Client Subnet in DNS Queries. RFC 7871.

[33] M. Cotton, L. Vegoda, R. Bonica, and B. Haberman. Special-Purpose IP Address Registries. RFC 6890.

[34] D. Dagon, N. Provos, C. P. Lee, and W. Lee. Corrupted DNS Resolution Paths: The Rise of a Malicious Resolution Authority. In *Networked and Distributed System Security Symposium (NDSS)*, 2008.

[35] A. Dainotti, C. Squarcella, E. Aben, K. C. Claffy, M. Chiesa, M. Russo, and A. Pescapé. Analysis of country-wide Internet outages caused by censorship. In *ACM Internet Measurement Conference (IMC)*, 2011.

[36] J. Dalek, B. Haselton, H. Noman, A. Senft, M. Crete-Nishihata, P. Gill, and R. J. Deibert. A Method for Identifying and Confirming the Use of URL Filtering Products for Censorship. In *ACM Internet Measurement Conference (IMC)*, 2013.

[37] N. Daswani and M. Stoppelman. The Anatomy of Clickbot.A. In *USENIX Workshop on Hot Topics in Understanding Botnets (HotBots)*, 2007.

[38] V. Dave, S. Guha, and Y. Zhang. Measuring and Fingerprinting Click-spam in Ad Networks. In *ACM SIGCOMM*, 2012.

[39] V. Dave, S. Guha, and Y. Zhang. ViceROI: Catching Click-Spam in Search Ad Networks. In *ACM Conference on Computer and Communications Security (CCS)*, 2013.

[40] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *USENIX Security Symposium (USENIX)*, 2004.

[41] D. Dittrich and E. Kenneally. The Menlo Report: Ethical Principles Guiding Information and Communication Technology Research. Technical report, U.S. Department of Homeland Security, Aug 2012.

[42] J. Dupre. What is Cloaking and Why Do Affiliate Marketers Use It? http://justindupre.com/what-is-cloaking-and-why-do-affiliate-marketers-use-it/, May 2010.

[43] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman. A Search Engine Backed by Internet-Wide Scanning. In *ACM Conference on Computer and Communications Security (CCS)*, 2015.

[44] Z. Durumeric, E. Wustrow, and J. A. Halderman. ZMap: Fast Internet-Wide Scanning and its Security Applications. In *USENIX Security Symposium*, 2013.

[45] B. G. Edelman. Google Click Fraud Inflates Conversion Rates and Tricks Advertisers into Overpaying. http://www.benedelman.org/news/011210-1.html, Jan. 2010.

[46] R. Ensafi, D. Fifield, P. Winter, N. Feamster, N. Weaver, and V. Paxson. Examining How the Great Firewall Discovers Hidden Circumvention Servers. In *ACM Internet Measurement Conference (IMC)*, 2015.

[47] R. Ensafi, J. Knockel, G. Alexander, and J. R. Crandall. Detecting Intentional Packet Drops on the Internet via TCP/IP Side Channels. In *Workshop on Passive and Active Network Measurement (PAM)*, 2014.

[48] R. Ensafi, J. C. Park, D. Kapur, and J. R. Crandall. Idle Port Scanning and Non-interference Analysis of Network Protocol Stacks Using Model Checking. In *USENIX Security Symposium (USENIX)*, 2010.

[49] R. Ensafi, P. Winter, A. Mueen, and J. R. Crandall. Analyzing the Great Firewall of China Over Space and Time. *Privacy Enhancing Technologies Symposium (PETS)*, 2015.

[50] O. Farnan, A. Darer, and J. Wright. Poisoning the Well – Exploring the Great Firewall's Poisoned DNS Responses. In *ACM Workshop on Privacy in the Electronic Society (WPES)*, 2016.

[51] Federal Bureau of Investigation. International Cyber Ring That Infected Millions of Computers Dismantled. http://www.fbi.gov/news/stories/2011/november/malware_110911, Nov. 2011.

[52] D. Fifield and L. Tsai. Censors' Delay in Blocking Circumvention Proxies. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2016.

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

[54] 5 Ways to Protect Against the Ad Fraud Surge in Q4. https://www.whiteops.com/q4-ad-fraud-surge, Oct. 2017.

[55] Freedom House. Freedom on the Net. 2016.

[56] M. Giuliani. ZeroAccess, an advanced kernel mode rootkit. `http://www.prevx.com/blog/171/ZeroAccess-an-advanced-kernel-mode-rootkit.html`, Apr. 2011.

[57] The Go Programming Language. `https://golang.org/`.

[58] Google Ads: Ad Traffic Quality Resource Center Overview. `http://www.google.com/ads/adtrafficquality/`.

[59] Google Inc. About smart pricing. *AdWords Help*, Apr. 2013.

[60] Google Inc. Google Services Agreement for InfoSpace LLC. `http://www.sec.gov/Archives/edgar/data/1068875/000119312514121780/d702452dex101.htm`, March 2014.

[61] Google Inc. How Google uses conversion data. *AdWords Help*, May 2014.

[62] Google Public DNS. `https://developers.google.com/speed/public-dns/`.

[63] T. Greene. ZeroAccess bot-herders abandon click-fraud network. `http://www.networkworld.com/news/2013/121913-zeroaccess-277113.html`, Dec. 2013.

[64] C. Grier et al. Manufacturing Compromise: The Emergence of Exploit-as-a-Service. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, October 2012.

[65] R. Gummadi, H. Balakrishnan, P. Maniatis, and S. Ratnasamy. Not-a-Bot: Improving Service Availability in the Face of Botnet Attacks. In *Usenix Symposium on Networked Systems Design and Implementation (NSDI)*, 2009.

[66] H. Haddadi. Fighting Online Click-Fraud Using Bluff Ads. *ACM SIGCOMM Computer Communication Review (CCR)*, 2010.

[67] T. Holz, M. Steiner, F. Dahl, E. Biersack, and F. Freiling. Measurements and Mitigation of Peer-to-Peer-based Botnets: A Case Study on Storm Worm. In *USENIX Conference on Large-scale Exploits and Emergent Threats (LEET)*, 2008.

[68] F. Howard. Exploring the Blackhole exploit kit. `http://nakedsecurity.sophos.com/exploring-the-blackhole-exploit-kit/`.

[69] HSI seizes Silk Road underground black market website. `http://www.ice.gov/news/releases/1310/131002baltimore.htm`, 2013.

[70] B. Huffaker, M. Fomenkov, and k. claffy. Geocompare: a comparison of public and commercial geolocation databases - Technical Report . Technical report, Cooperative Association for Internet Data Analysis (CAIDA), May 2011.

[71] ICLab. ICLab: a Censorship Measurement Platform. `https://iclab.org/`.

[72] P. Ipeirotis. Uncovering an advertising fraud scheme. Or "the Internet is for porn". `http://www.behind-the-enemy-lines.com/2011/03/uncovering-advertising-fraud-scheme.html`, Mar. 2011.

[73] J. Jiang, J. Liang, K. Li, J. Li, H. Duan, and J. Wu. Ghost Domain Name: Revoked yet Still Resolvable. In *Networked and Distributed System Security Symposium (NDSS)*, 2012.

[74] J. P. John, A. Moshchuk, S. D. Gribble, and A. Krishnamurthy. Studying Spamming Botnets Using Botlab. In *Usenix Symposium on Networked Systems Design and Implementation (NSDI)*, 2009.

[75] B. Jones, N. Feamster, V. Paxson, N. Weaver, and M. Allman. Detecting DNS Root Manipulation. In *Workshop on Passive and Active Network Measurement (PAM)*, 2016.

[76] B. Jones, T.-W. Lee, N. Feamster, and P. Gill. Automated Detection and Fingerprinting of Censorship Block Pages. In *ACM Internet Measurement Conference (IMC)*, 2014.

[77] A. Juels, S. Stamm, and M. Jakobsson. Combating Click Fraud Via Premium Clicks. In *USENIX Security Symposium (USENIX)*, 2007.

[78] J. Jung, V. Paxson, A. W. Berger, and H. Balakrishnan. Fast Portscan Detection Using Sequential Hypothesis Testing. In *IEEE Symposium on Security and Privacy (S&P)*, 2004.

[79] Kaffeine. MagicTraffic : a look inside a Zaccess/Sirefef affiliate. `http://malware.dontneedcoffee.com/2013/11/magictraffic-look-inside-zaccesssirefef.html`, 2013.

[80] H. Kang, K. Wang, D. Soukal, F. Behr, and Z. Zheng. Large-scale Bot Detection for Search Engines. In *International Conference on World Wide Web (WWW)*, 2010.

[81] S. Khattak, D. Fifield, S. Afroz, M. Javed, S. Sundaresan, V. Paxson, S. J. Murdoch, and D. McCoy. Do You See What I See? Differential Treatment of Anonymous Users. In *Networked and Distributed System Security Symposium (NDSS)*, 2016.

[82] S. Khattak, M. Javed, P. D. Anderson, and V. Paxson. Towards Illuminating a Censorship Monitor's Model to Facilitate Evasion. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2013.

[83] L. Kim. The Most Expensive Keywords in Google AdWords. `http://www.wordstream.com/blog/ws/2011/07/18/most-expensive-google-adwords-keywords`, July 2011.

[84] C. Kintana, D. Turner, J.-Y. Pan, A. Metwally, N. Daswani, E. Chin, and A. Bortz. The Goals and Challenges of Click Fraud Penetration Testing Systems. In *International Symposium on Software Reliability Engineering (ISSRE)*, 2009.

[85] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The Click Modular Router. *ACM Transactions Computer Systems*, August 2000.

[86] B. Krebs. Security bug in dell pcs shipped since 8/15. `https://krebsonsecurity.com/2015/11/security-bug-in-dell-pcs-shipped-since-815/`.

[87] B. Krebs. Takedowns: The Shuns and Stuns That Take the Fight to the Enemy. In *McAfee Security Journal*, volume 6, 2010.

[88] B. Krebs. Fake Antivirus Industry Down, But Not Out. `http://krebsonsecurity.com/2011/08/fake-antivirus-industry-down-but-not-out/`, August 2011.

[89] B. Krebs. Reports: Liberty Reserve Founder Arrested, Site Shuttered. `http://krebsonsecurity.com/2013/05/reports-liberty-reserve-founder-arrested-site-shuttered/`, 2013.

[90] B. Krebs. ZeroAccess Botnet Down, But Not Out. `http://krebsonsecurity.com/tag/zeroaccess-takedown/`, Dec. 2013.

[91] C. Kreibich, N. Weaver, C. Kanich, W. Cui, and V. Paxson. GQ: Practical Containment for Measuring Modern Malware Systems. In *ACM Internet Measurement Conference (IMC)*, 2011.

[92] N. Kshetri. The Economics of Click Fraud. *IEEE Symposium on Security and Privacy (S&P)*, 2010.

[93] M. Kührer, T. Hupperich, J. Bushart, C. Rossow, and T. Holz. Going Wild: Large-Scale Classification of Open DNS Resolvers. In *ACM Internet Measurement Conference (IMC)*, 2015.

[94] M. Kührer, T. Hupperich, C. Rossow, and T. Holz. Exit from Hell? Reducing the Impact of Amplification DDoS Attacks. In *USENIX Security Symposium*, 2014.

[95] K. Levchenko, A. Pitsillidis, N. Chachra, B. Enright, M. Félegyházi, C. Grier, T. Halvorson, C. Kanich, C. Kreibich, H. Liu, D. McCoy, N. Weaver, V. Paxson, G. M. Voelker, and S. Savage. Click Trajectories: End-to-End Analysis of the Spam Value Chain. In *IEEE Symposium on Security and Privacy (S&P)*, 2011.

[96] The Lote Clicking Agent. `http://www.clickingagent.com/`.

[97] G. Lowe, P. Winters, and M. L. Marcus. The Great DNS Wall of China. Technical report, New York University, 2007.

[98] MaxMind. `https://www.maxmind.com/`.

[99] D. McCoy, H. Dharmdasani, C. Kreibich, G. M. Voelker, and S. Savage. Priceless: The Role of Payments in Abuse-advertised Goods. In *ACM Conference on Computer and Communications Security (CCS)*, 2012.

[100] K. McNamee. Malware Analysis Report. Botnet: ZeroAccess/Sirefef. `http://www.kindsight.net/sites/default/files/Kindsight_Malware_Analysis-ZeroAcess-Botnet-final.pdf`, February 2012.

[101] A. Metwally, D. Agrawal, and A. El Abbadi. DETECTIVES: DETEcting Coalition hiT Inflation attacks in adVertising nEtworks Streams. In *International Conference on World Wide Web (WWW)*, 2007.

[102] A. Metwally, F. Emekçi, D. Agrawal, and A. El Abbadi. SLEUTH: Single-pubLisher attack dEtection Using correlaTion Hunting. In *Conference on Very Large Data Bases (VLDB)*, 2008.

[103] Microsoft, Yahoo! Change Search Landscape. `http://www.microsoft.com/en-us/news/press/2009/jul09/07-29release.aspx`, July 2009.

[104] Microsoft Corporation. Zeroaccess botnet Legal Notice. `https://www.botnetlegalnotice.com/zeroaccess/`.

[105] B. Miller, P. Pearce, C. Grier, C. Kreibich, and V. Paxson. What's Clicking What? Techniques and Innovations of Todays Clickbots. In *Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, 2011.

[106] S. Mitchell and B. Collins. Porn blocking: What the Big Four ISPs Actually Did. `http://www.alphr.com/networking/20643/porn-blocking-what-the-big-four-isps-actually-did`, August 2015.

[107] D. Moore, G. Voelker, and S. Savage. Inferring Internet Denial-of-Service Activity. In *USENIX Security Symposium (USENIX)*, 2001.

[108] Z. Nabi. The Anatomy of Web Censorship in Pakistan. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2013.

[109] A. Neville and R. Gibb. ZeroAccess Indepth (Symantec Corporation White Paper). `http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/zeroaccess_indepth.pdf`, October 2013.

[110] Q. Northon. China Blocks LiveJournal. https://www.wired.com/2007/03/china-blocks-livejournal/, March 2007.

[111] Ooniprobe: A Network Interference Detection Tool. https://github.com/thetorproject/ooni-probe.

[112] OpenNet Initiative. https://opennet.net/.

[113] Open Resolver Project. http://openresolverproject.org/.

[114] J. C. Park and J. R. Crandall. Empirical Study of a National-Scale Distributed Intrusion Detection System: Backbone-Level Filtering of HTML Responses in China. In *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2010.

[115] P. Pearce, V. Dave, C. Grier, K. Levchenko, S. Guha, D. McCoy, V. Paxson, S. Savage, and G. M. Voelker. Characterizing Large-Scale Click Fraud in ZeroAccess. In *ACM Conference on Computer and Communications Security (CCS)*, 2014.

[116] P. Pearce, R. Ensafi, F. Li, N. Feamster, and V. Paxson. Augur: Internet-Wide Detection of Connectivity Disruptions. In *IEEE Symposium on Security and Privacy (S&P)*, 2017.

[117] P. Pearce, R. Ensafi, F. Li, N. Feamster, and V. Paxson. Towards Continual Measurement of Global Network-Level Censorship. In *IEEE Security & Privacy Magazine, Special Issue*, 2018.

[118] P. Pearce, C. Grier, V. Paxson, V. Dave, D. McCoy, G. M. Voelker, and S. Savage. The ZeroAccess Auto-Clicking and Search-Hijacking Click Fraud Modules. Technical report, EECS Department, University of California, Berkeley, Dec 2013.

[119] P. Pearce, B. Jones, F. Li, R. Ensafi, N. Feamster, N. Weaver, and V. Paxson. Global Measurement of DNS Manipulation. In *USENIX Security Symposium (USENIX)*, 2017.

[120] P. Pearce, B. Jones, F. Li, R. Ensafi, N. Feamster, N. Weaver, and V. Paxson. Global Measurement of DNS Manipulation. In *USENIX ;login:*, Winter 2017.

[121] M. Polychronakis, P. Mavrommatis, and N. Provos. Ghost turns Zombie: Exploring the Life Cycle of Web-based Malware. In *USENIX Conference on Large-scale Exploits and Emergent Threats (LEET)*, 2008.

[122] Pricewaterhouse Coopers. IAB Internet Advertising Revenue Report: 2017 Full Year Results. https://www.iab.com/wp-content/uploads/2018/05/IAB-2017-Full-Year-Internet-Advertising-Revenue-Report.REV2_.pdf, May 2018.

[123] Z. Qian, Z. M. Mao, Y. Xie, and F. Yu. Investigation of triangular spamming: A stealthy and efficient spamming technique. In *IEEE Symposium on Security and Privacy (S&P)*, 2010.

[124] A. Razaghpanah, A. Li, A. Filastò, R. Nithyanand, V. Ververis, W. Scott, and P. Gill. Exploring the Design Space of Longitudinal Censorship Measurement Platforms. Technical Report 1606.01979, ArXiv CoRR, 2016.

[125] E. Rodionov and A. Matrosov. The Evolution of TDL: Conquering x64. `http://go.eset.com/us/resources/white-papers/The_Evolution_of_TDL.pdf`, 2011.

[126] C. Rossow, D. Andriesse, T. Werner, B. Stone-Gross, D. Plohmann, C. J. Dietrich, and H. Bos. SoK: P2PWNED — Modeling and Evaluating the Resilience of Peer-to-Peer Botnets. In *IEEE Symposium on Security and Privacy (S&P)*, May 2013.

[127] M. Salganik. Bit by Bit: Social Research for the Digital Age. `http://www.bitbybitbook.com/`, 2016.

[128] Sam Burnett and Nick Feamster. Encore: Lightweight Measurement of Web Censorship with Cross-Origin Requests. In *ACM SIGCOMM*, 2015.

[129] K. Schomp, T. Callahan, M. Rabinovich, and M. Allman. On Measuring the Client-Side DNS Infrastructure. In *ACM Internet Measurement Conference (IMC)*, 2013.

[130] W. Scott, T. Anderson, T. Kohno, and A. Krishnamurthy. Satellite: Joint Analysis of CDNs and Network-Level Interference. In *USENIX Annual Technical Conference (ATC)*, 2016.

[131] A. Sfakianakis, E. Athanasopoulos, and S. Ioannidis. CensMon: A Web Censorship Monitor. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2011.

[132] L. Sinclair. Click fraud rampant in online ads, says Bing. `http://www.theaustralian.com.au/media/click-fraud-rampant-in-online-ads-says-bing/story-e6frg996-1226056349034`, May 2011.

[133] A. Spence. Russia Accuses Latvia of "blatant censorship" after Sputnik News Site is Shut Down. `https://goo.gl/wIUUCr`, Mar. 2016.

[134] The Tor Project. OONI: Open observatory of network interference. `https://ooni.torproject.org/`.

[135] K. Thomas, E. Bursztein, C. Grier, G. Ho, N. Jagpal, A. Kapravelos, D. McCoy, A. Nappa, V. Paxson, P. Pearce, N. Provos, and M. A. Rajab. Ad Injection at Scale: Assessing Deceptive Advertisement Modifications. In *IEEE Symposium on Security and Privacy (S&P)*, 2015.

[136] K. Thomas, D. Huang, D. Wang, E. Bursztein, C. Grier, T. J. Holt, C. Kruegel, D. McCoy, S. Savage, and G. Vigna. Framing dependencies introduced by underground commoditization. In *Workshop on the Economics of Information Security*, 2015.

[137] The Tor Project. https://www.torproject.org/.

[138] Transmission Control Protocol. RFC 793, Sept. 1981.

[139] M. C. Tschantz, S. Afroz, Anonymous, and V. Paxson. SoK: Towards Grounding Censorship Circumvention in Empiricism. In *IEEE Symposium on Security and Privacy*, 2016.

[140] G. Tuysuz and I. Watson. Turkey Blocks YouTube Days after Twitter Crackdown. http://www.cnn.com/2014/03/27/world/europe/turkey-youtube-blocked/, Mar. 2014.

[141] A. Tuzhilin. The Lane's Gifts v. Google Report. http://googleblog.blogspot.com/pdf/Tuzhilin_Report.pdf, 2005.

[142] N. Villeneuve. Koobface: Inside a Crimeware Network. http://www.infowar-monitor.net/reports/iwm-koobface.pdf, Nov. 2010.

[143] J. Wakefield. Bbc news: Lenovo taken to task over 'malicious' adware. https://www.bbc.com/news/technology-31533028.

[144] N. Weaver, C. Kreibich, and V. Paxson. Redirecting DNS for Ads and Profit. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2011.

[145] White Ops. The Methbot Operation. https://www.whiteops.com/methbot.

[146] S. Wilson. The Logic of Russian Internet Censorship. https://www.washingtonpost.com/news/monkey-cag>e/wp/2014/03/16/the-logic-of-russian-internet-censorship/, Mar. 2014.

[147] P. Winter. The Philippines are blocking Tor? Tor Trac ticket, June 2012. https://bugs.torproject.org/6258.

[148] P. Winter and S. Lindskog. How the Great Firewall of China is Blocking Tor. In *USENIX Workshop on Free and Open Communications on the Internet (FOCI)*, 2012.

[149] J. Wyke. The ZeroAccess Botnet: Mining and Fraud for Massive Financial Gain. http://www.sophos.com/en-us/why-sophos/our-people/technical-papers/zeroaccess-botnet.aspx, September 2012.

[150] J. Wyke. ZeroAccess. http://www.sophos.com/en-us/why-sophos/our-people/technical-papers/zeroaccess.aspx, 2012.

[151] X. Xu, Z. M. Mao, and J. A. Halderman. Internet Censorship in China: Where Does the Filtering Occur? In *Workshop on Passive and Active Network Measurement (PAM)*, 2011.

[152] F. Yu, Y. Xie, and Q. Ke. SBotMiner: Large Scale Search Bot Detection. In *Conference on Web Search and Data Mining (WSDM)*, 2010.

[153] X. Zhang, J. Knockel, and J. R. Crandall. Original SYN: Finding machines hidden behind firewalls. In *IEEE Conference on Computer Communications (INFOCOM)*, 2015.

[154] J. Zittrain and B. G. Edelman. Internet filtering in China. *IEEE Internet Computing*, 2003.