# Measuring and Improving Security and Privacy on the Web: Case Studies with QR Codes, Third-Party Tracking, and Archives

Ada Lerner

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2017

Reading Committee:

Franziska Roesner, Chair

Tadayoshi Kohno, Chair

Arvind Krishnamurthy

Program Authorized to Offer Degree:
Paul G. Allen School of Computer Science and Engineering

University of Washington

**Abstract**

Measuring and Improving Security and Privacy on the Web:
Case Studies with QR Codes, Third-Party Tracking, and Archives

Ada Lerner

Co-Chairs of the Supervisory Committee:
Assistant Professor Franziska Roesner
Paul G. Allen School of Computer Science and Engineering

Short-Dooley Professor Tadayoshi Kohno
Paul G. Allen School of Computer Science and Engineering

The web is deeply integrated into the economic, governmental, and social fabrics of our society, and it promises to help fulfill people's diverse and critical needs more easily, cheaper, faster, and more democratically. However, to fulfill this promise, we must study the security and privacy implications of the web and its surrounding technologies, and ensure that security and privacy are incorporated into their future evolution. In this dissertation, I present measurement and analysis work which studies the web and a set of the web's neighboring "sister technologies" — QR codes and web archives — forming insights and foundations for the ways that we can make the web more secure and privacy preserving. I identify and quantify current, past, and future properties of the web which are critical to its ability to provide security and privacy to its users, and synthesize lessons from these measurements and analyses which aim to guide the future designers and regulators of technologies surrounding the web in ensuring that it serves the needs of all who use it.

# TABLE OF CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# ACKNOWLEDGMENTS

This dissertation has been in development my whole time at UW CSE, and it is the accumulation of six years of learning which I have been fortunate enough to share with so many others.

I must first acknowledge, thank, and laud my advisors, Franziska Roesner and Tadayoshi Kohno, who have dedicated their hearts and minds to being everything advisors can and should be. They have been unrelentingly positive, reminding me constantly that the work we have done together is exciting and worth doing, and they have consistently offered me the challenges and support I have needed to grow as a researcher, a teacher, and a person. Throughout my PhD, they valued not only my research, but also my mental health, my happiness, my identity, my career goals, my family, and my life. They are my mentors and my friends, and I am glad every day that we will be colleagues in the community of computer security and privacy for a long time. Thank you both.

I thank also the advisors of my early years, Arvind Krishnamurthy and Thomas Anderson, who brought me into graduate school and who set me on the first steps of this path I walk every day. The networks lab has been a second home to me in CSE. I am often thankful for their belief in me, their advice and mentorship, and their continued support, even as I moved away, into the security lab. Thanks in particular to Arvind for serving on my committee.

Many thanks to my final committee member, David McDonald. At my General Exam, David offered insight and aid regarding my research proposal, helping to develop it in thoughtful, realistic, and challenging directions. Ever since then, when I consider a new research direction, I think of the questions he asked at that exam. His support will bolster me throughout my career.

None of this work would have been possible without the collaborators and co-authors of my dissertation work, Tadayoshi Kohno, Kirk Ouimet, Franziska Roesner, Alisha Saxena, Anna Kornfeld Simpson, Ben Turley, and Anthony Vance, each of whom made this work possible and lent innumerable insights to it. I thank also my collaborators from the other papers I co-authored during my PhD: Tamara Denning, Adam Shostack, Emily McReynolds, Will Scott, and Eric Zeng. Your contributions to the work we did together helped me develop as a writer, a computer scientist, and a security and privacy researcher.

It takes a village to raise a PhD, and so I thank members both past and present of the UW CSE Security and Privacy Research Lab. Our lab is a place where I have received constant advice, feedback, support, and friendship. These people include Camille Cobb, Alexei Czeskis, Tamara Denning, Miro Enev, Gennie Gebhart, Karl Koscher, Kiron Lebeck, Shrirang Mare, Yutaka Matsubara, Peter Ney, Temitope Oluwafemi, Mitali Palekar, Kimberly Ruth, Alisha Saxena, Lucy Simko, Anna Kornfeld Simpson, Ian Smith, Alex Takakuwa, Paul Vines, and Eric Zeng. It has been a privilege to work with all of you, as well as a delight to play soccer and softball and board games together, to make up silly Pocsci ideas and custom emoji. The community of our lab is strong, thoughtful, inclusive, and creative, and by creating that culture, each of these people has contributed to making this the best place I can imagine to have done the work of a PhD.

I also thank a variety of other people from UW CSE at large, with whom I have had many discussions, debates, delights, and discourses: Katelin Bailey, Catherine Baker, Aaron Bauer, Adam Blank, Sam Castle, Raymond Cheng, Lilian de Greef, Kira Goldner, Dan Grossman, Daniel Halperin, Seungyeop Han, Tomas Isdal, Eunice Jun, Richard Ladner, Niel Lebeck, Hank Levy, Jialin Li, Yun-En Liu, Leah Perlmutter, Simon Peter, Dan Ports, Hannah Rashkin, Katharina Reinecke, Talia Ringer, Maarten Sap, Naveen Kr. Sharma, Sam Sudar, Adriana Szekeres, Steve Tanimoto, Zachary Tatlock, Xiao Sophia Wang, Luke Zettlemoyer, Irene Zhang, and Li Zilles.

I have also had the pleasure of being a part of the UW Tech Policy Lab since its earliest days. Thank you to everyone from the discussion group who has enriched my understanding of the world through the lens of law. A special thanks to Emily McReynolds, who has been not only a colleague, a mentor, and a provider of delicious banh mi, but also a friend. And many thanks as well to Ryan Calo, who drew me into the tech policy world through his Robotics Law and Policy class.

Lindsay Michimoto and Elise DeGoede Dorough were wonderful graduate program advisors, whose service strengthened, validated, and supported me throughout my time in the program. Elise in particular has been a powerful advocate, often taking thoughtful, effective action in situations where others might have left the work for someone else.

My gratefulness extends past UW of course. Thank you as well to Barath Raghavan, who was a wonderful mentor during my internship at De Novo Group, and off whom I have always felt comfortable bouncing weird ideas. May your banana trees grow tall. And it was a pleasure as well to work with Giulia Fanti, with whom I hope I will have many chances to collaborate in the future. Thank you as well to Jesus Garcia, who entrusted himself to me in my first real mentorship opportunity, and to Yahel Ben David for his enthusiasm and the opportunity for my internship.

I didn't spring out of thin air, a graduate student in computer science, and so I thank those who supported me before UW CSE. I am grateful to the computer science faculty of Amherst College, and especially to John Rager, who has always had an open door and an open ear, and to Scott Kaplan, who first introduced me to research and has remained a trusted mentor to this day. I'm excited to be starting a career as colleagues with them in the small community of liberal arts computer scientists.

Finally, thank you to my mother Katlyn, my father, Ken, and my sister Kara, for your belief in me, and for your love. Your lifelong support of my education and your nurturing of my kindness have brought me where I am today. And I feel so grateful for the rest

of my family — Lerner, Palevsky, Roseman, Stover, Tomlins, Wiatrak, Wood — who have surrounded me with patient, supporting, accepting love all my life. And last, thank you to Hilary, who has been my teammate and partner through this journey, and to Chai, my cat, who has been present on my lap as I did a large fraction of the work of this PhD.

The work of my PhD and of this dissertation was supported in part by NSF Awards CNS-0846065, IIS-1302709, CNS-1463968, and CNS-1513575, and by the University of Washington Tech Policy Lab, which was founded with a gift from Microsoft in 2013.

# DEDICATION

To my family in all its breadth, who have helped me thrive.

# Chapter 1

# INTRODUCTION

People today use the web in critical aspects of their lives, making the security and privacy of the web critical in both personally and public ways. For example, personally sensitive uses of the web include online dating, purchasing sensitive items, and communication with friends, partners, and relatives. Meanwhile, important public uses include of the web include obtaining political news, registering to vote, and the exercise of free speech. This trend offers promises of affordability, democratization, and freedom for many important aspects of our lives, but it also comes with great risks: by networking our private matters and storing private details on remote servers, we risk privacy violations which were impossible or unthinkable when those matters were physically isolated behind closed doors. And by putting our most important public matters online, we could be endangering the integrity of our democracy and our free society. To fulfill these promises and mitigate these risks, we will need to deeply understand the security and privacy of the web as it evolves, and incorporate that understanding into the way we design and regulate the web going forward.

This dissertation uses measurement and analysis to study the web and a set of the web's neighboring "sister technologies", forming insights and foundations for the ways that we can make the web more secure and privacy preserving. Its primary approach is through measurement of the web and of those neighboring technologies, measurements which quantify our interactions with the web and the risks involved in those interactions. In particular, this dissertation presents new measurement techniques for examining the web longitudinally, new measurements of how third-party web tracking has evolved over time, the development and quantification of attacks and defenses against user views of web archives, and the first large-scale measurement study of the use of QR codes and their interaction with the web.

Throughout, I synthesize lessons, both for technologists and policymakers, which provide guidance on effective directions for integrating security and privacy into the design and regulation of technology to make us safer and stronger as individuals and a broader society.

## *1.1 Themes and Challenges*

The primary themes and challenges of this dissertation arise from its use of novel measurement techniques and its study of the way that the web interacts with what I will term "sister technologies": web-adjacent technologies that interact with, enhance, compliment, and are complimented by the web.

### *Novel Measurement Techniques*

New insights about technologies like the web sometimes require new approaches and unique datasets. Acting on this observation, Chapters 2 and 3 describe, develop, and use web archives as a tool and subject of measurement study, while Chapter 4 performs a unique study of global QR code usage through the use of a dataset of 87 million barcode scans obtained through collaboration with industry. For example, I observe that longitudinal measurements of the web are valuable for their ability to depict trends, but challenging, since the web is constantly changing. Chapter 2 overcomes this challenge by developing a new method of using archival data to perform longitudinal measurements over nearly the whole history of the web. Taking another approach to new measurements, Chapter 4 uses a unique dataset, obtained through collaboration with industry, to perform the first large-scale academic study of the use of QR (Quick Response) codes in the wild. Through these new measurement techniques, this dissertation demonstrates the power of taking new approaches to measure what is and has been, and it offers insights into the real security and privacy threats we face today and tomorrow. By seeing how those threats have changed and will change, it guides us in integrating security and privacy into the design, implementation, and regulation of the web and its sister technologies.

*The Interaction of the Web With Its Sister Technologies*

The second major theme of this dissertation is the interaction of the web with related technologies that refer to, augment, and compliment the web. This dissertation specifically considers the following sister technologies: third-party web tracking, web archives, and QR codes. Each enables unique interactions with the web. For example, QR codes enable pointers to the web to exist in the physical world, while web archives allow us to time travel through the web, studying and citing its past. Additionally, these sister technologies sometimes fundamentally shape the web, as with web tracking, which enables much of the personalized advertising which underlies the economy of the web, allowing many of our most powerful and useful web services to exist, while also pushing the commercial structure and incentives of the web in a direction that emphasizes the value of individuals' personal data. Given the technical and economic importance of these sister technologies, this dissertation studies them alongside the web in order to understand the way that web security and privacy can be made to serve peoples' needs.

## *1.2 Contributions*

This dissertation is presented in three chapters, each of which measures and analyses either the web itself, or certain technologies which surround, expand, and enhance the web. Thus the topics of the following three chapters are measurements of the evolution of third-party tracking on the web; the security of web archives against malicious manipulation; and the expansive ecosystem of QR code usage and its relation to the web. Web archives appear in both Chapter 2 and Chapter 3: in the first case, they are used to enable a new longitudinal measurement technique, while in the second they become a topic of study themselves. Finally, QR codes appear in Chapter 4, as a perspective from which I explore the ways in which emerging uses of technology piggyback upon and differ from the web and other modern technologies.

### *1.2.1 Using Archival Sources For Longitudinal Measurements Of Third Party Web Tracking*

Chapter 2 focuses on the development of a new measurement technique for longitudinal measurements of the web, as well as the application of that technique to one important privacy concern on the web: third-party web tracking. The primary contributions of this chapter include the idea of measuring technical properties of the web longitudinally, using web archives; the quantification and evaluation of the challenges of using archival data in longitudinal measurements; the development of a tool, TrackingExcavator, which performs these measurements automatically; and the presentation and interpretation of data depicting 20 years of evolution of the history of third-party web tracking, using the above tool and techniques. This chapter shows that tracking has increased dramatically since the earliest days of the web: not only have the number of trackers increased dramatically, but their power has also increased, with some trackers individually able to follow people across more than 20% of the most popular 500 sites on the web, and some third parties appearing on over 40% of the most popular 500 sites in some years. I also made public the data from this study, its analysis code and TrackingExcavator itself, allowing other researchers to longitudinally measure a variety of properties of the web far beyond web tracking.

### *1.2.2 Rewriting History: Changing the Archived Web From the Present*

In Chapter 3, the dissertation pivots from considering web archives as a source of data for longitudinal web measurement to considering them as a subject of study themselves. Observing that web archives are often cited in socially important contexts such as journalism, scientific articles, and legal proceedings, I observe that such important uses may motivate adversaries to attempt to manipulate archives for their own purposes. This dissertation is the first to our knowledge to study the ways in which the Wayback Machine, perhaps the largest modern web archive, is vulnerable to adversarial manipulation of the ways that clients view its archival content. I discover a variety of vulnerabilities which expose client views to manipulation, and develop attacks which exploit those vulnerabilities. I then measure the

prevalence of those vulnerabilities, finding that they are quite common in practice, exposing a significant fraction of the archived web to adversarial manipulation: nearly three-quarters of popular snapshots from recent years are vulnerable to complete takeover by at least one adversary. Finally, I consider the defenses that may be deployed in order to protect client views of archived websites and increase the trust that people who rely on archival data for socially important uses can put in that data.

### *1.2.3 Analyzing the Use of Quick Response Codes in the Wild*

Chapter 4 of this dissertation presents measurements of the use of QR codes by real uses across the world. This chapter is to our knowledge the first large-scale academic analysis of the use of QR codes in the wild. It presents data showing the wide variety of ways people use QR codes. From that data, I derive a series of lessons for developers and users of technologies like QR codes. These lessons are expansive, covering both modern technology like QR codes, as well as considering QR codes as a proxy for future emerging technologies which embed information in the environment. Additionally, I observe that the QR codes in our dataset are dominantly web addresses (87% of scans), illustrating QR codes role as a sister technology to the web. Despite the importance of web related uses, we also identify a variety of interesting niche uses, such as an extremely popularly scanned code for donating bitcoins to The Pirate Bay, as well as a number of malicious Android apps and other malware distribution sites among the codes. This chapter considers the ways in which the physicality and mobile nature of QR codes may bear on security and privacy for the web as it is used through QR codes.

### *1.2.4 Summary of Contributions*

Together, these chapters provide a set of perspectives that can ground future approaches to understanding and measuring security and privacy challenges on the web, and to designing, implementing, and regulating technologies that help the web be a safer, more beneficial system for all people.

# Chapter 2

# USING ARCHIVAL SOURCES FOR LONGITUDINAL MEASUREMENTS OF THIRD PARTY WEB TRACKING

In this chapter, I engage with both of the themes of this dissertation by presenting a new measurement technique (longitudinal web measurement) which is enabled by a sister technology (web archives). Specifically, I develop a technique to perform longitudinal web measurements, implement that technique in a tool called TrackingExcavator, and demonstrate the technique by using TrackingExcavator to perform a study of changes in third-party web tracking over 20 years. Finally, observe that TrackingExcavator can be used for a wide variety of measurements using web archives, and thus will reappear as the tool used for the measurements in Chapter 3.

The work of this chapter previously appeared in a 2016 paper [14], and citations of this work should refer to that paper.

## *2.1 Introduction*

*Third-party web tracking* is the practice by which third parties like advertisers, social media widgets, and website analytics engines — embedded in the first party sites that users visit directly — re-identify users across domains as they browse the web. Web tracking, and the associated privacy concerns from tracking companies building a list of sites users have browsed to, has inspired a significant and growing body of academic work in the computer security and privacy community, attempting to understand, measure, and defend against such tracking (e.g., [147, 100, 101, 72, 103, 63, 80, 82, 83, 102, 49, 50, 51, 67, 43, 86, 177, 12, 134, 13, 175, 106, 69, 169, 68, 23, 64, 164, 56, 149, 39, 99, 19, 124, 107, 165]).

However, the research community's interest in web tracking comes relatively recently in

the history of web. To our knowledge, the earliest measurement studies began in 2005 [102], with most coming after 2009 — while display advertising and the HTTP cookie standard date to the mid-1990s [112, 104]. Though numerous studies have now been done, they typically consist of short-term measurements of specific tracking techniques. We argue that public and private discussions surrounding web tracking — happening in technical, legal, and policy arenas (e.g., [120, 179]) — ought to be informed not just by a single snapshot of the web tracking ecosystem but by a comprehensive knowledge of its trajectory over time. We provide such a comprehensive view in this chapter, conducting a measurement study of third-party web tracking across 20 years since 1996.

Measurement studies of web tracking are critical to provide transparency for users, technologists, policymakers, and even those sites that include trackers, to help them understand how user data is collected and used, to enable informed decisions about privacy, and to incentivize companies to consider privacy. However, the web tracking ecosystem is continuously evolving, and others have shown that web privacy studies at a single point in time may only temporarily reduce the use of specific controversial tracking techniques [161]. While one can study tracking longitudinally starting in the present, as we and others have (e.g., [161, 102]), ideally any future developments in the web tracking ecosystem can be contextualized in a comprehensive view of that ecosystem over time — i.e., since the very earliest instance of tracking on the web. We provide that longitudinal, historical context in this chapter, asking: how has the third-party web tracking ecosystem evolved since its beginnings?

To answer this question, we apply a key insight: the Internet Archive's *Wayback Machine* [75] enables a retrospective analysis of third-party tracking on the web over time. The Wayback Machine[1] contains archives of full webpages, including JavaScript, stylesheets, and embedded resources, dating back to 1996. To leverage this archive, we design and implement a retrospective tracking detection and analysis platform called *TrackingExcavator* (Section 2.4), which allows us to conduct a longitudinal study of third-party tracking from

[1] `https://archive.org`

1996 to present (2016). TrackingExcavator logs in-browser behaviors related to web tracking, including: third-party requests, cookies attached to requests, cookies programmatically set by JavaScript, and the use of other relevant JavaScript APIs (e.g., HTML5 LocalStorage and APIs used in browser fingerprinting [43, 134], such as enumerating installed plugins). TrackingExcavator can run on both live as well as archived versions of websites.

Harnessing the power of the Wayback Machine for our analysis turns out to be surprisingly challenging (Section 2.6). Indeed, a key contribution of this chapter is our evaluation of the historical data provided by the Wayback Machine, and a set of lessons and techniques for extracting information about trends in third-party content over time. Through a comparison with ground truth datasets collected in 2011 (provided to us by the authors of [147]), 2013, 2015, and 2016, we find that the Wayback Machine's view of the past, as it relates to included third-party content, is imperfect for many reasons, including sites that were not archived due to `robots.txt` restrictions (which are respected by the Wayback Machine's crawlers), the Wayback Machine's occasional failure to archive embedded content, as well as site resources that were archived at different times than the top-level site. Though popular sites are typically archived at regular intervals, their embedded content (including third-party trackers) may thus be only partially represented. Whereas others have observed similar limitations with the Wayback Machine, especially as it relates to content visible on the top-level page [28, 88, 130], our analysis is focused on the technical impact of missing third-party elements, particularly with respect to tracking. Through our evaluation, we characterize what the Wayback Machine lets us measure about the embedded third parties, and showcase some techniques for best using the data it provides and working around some of its weaknesses (Section 2.6).

After evaluating the Wayback Machine's view into the past and developing best practices for using its data, we use TrackingExcavator to conduct a longitudinal study of the third-party web tracking ecosystem from 1996-2016 (Sections 2.5). We explore how this ecosystem has changed over time, including the prevalence of different web tracking behaviors, the identities and scope of popular trackers, and the complexity of relationships within the

ecosystem. Among our findings, we identify the earliest tracker in our dataset in 1996 and observe the rise and fall of important players in the ecosystem (e.g., the rise of Google Analytics to appear on over a third of all popular websites). We find that websites contact an increasing number of third parties over time (about 5% of the 500 most popular sites contacted at least 5 separate third parties in early 2000s, whereas nearly 40% do so in 2016) and that the top trackers can track users across an increasing percentage of the web's most popular sites. We also find that tracking behaviors changed over time, e.g., that third-party popups peaked in the mid-2000s and that the fraction of trackers that rely on referrals from other trackers has recently risen.

Taken together, our findings show that third-party web tracking is a rapidly growing practice in an increasingly complex ecosystem — suggesting that users' and policymakers' concerns about privacy require sustained, and perhaps increasing, attention. Our results provide hitherto unavailable historical context for today's technical and policy discussions.

In summary, our contributions are:

1. TrackingExcavator, a **measurement infrastructure** for detecting and analyzing third-party web tracking behaviors in the present and — leveraging the Wayback Machine — in the past (Section 2.4).
2. An **in-depth analysis** of the scope and accuracy of the Wayback Machine's view of historical web tracking behaviors and trends, and techniques for working around its weaknesses (Section 2.6).
3. A **longitudinal measurement study** of third-party cookie-based web tracking from 1996 to present (2016) — to the best of our knowledge, the longest longitudinal study of tracking to date (Section 2.5).

### *2.2 Background and Motivation*

Third-party web tracking is the practice by which entities ("trackers") embedded in webpages re-identify users as they browse the web, collecting information about the websites that they visit [147, 122]. Tracking is typically done for the purposes of website analytics, targeted ad-

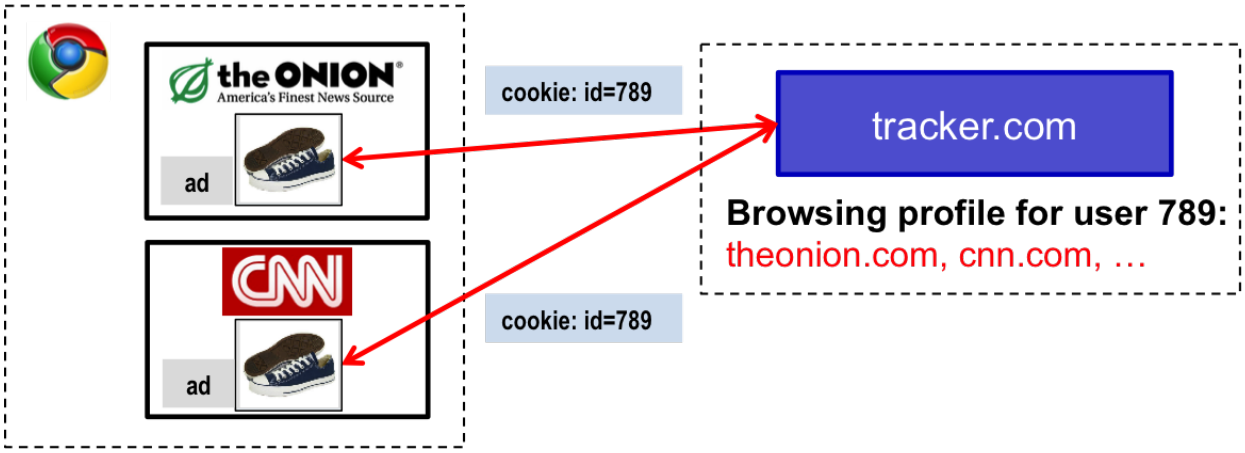



Figure 2.1: Overview of basic cookie-based web tracking. The third-party domain `tracker.com` uses a browser cookie to re-identify users on sites that embed content from `tracker.com`. This example shows *vanilla* tracking according to the taxonomy from [147]; other behaviors are described in Section 2.4.

vertising, and other forms of personalization (e.g., social media content). For example, when a user visits `www.cnn.com`, the browser may make additional requests to `doubleclick.net` to load targeted ads and to `facebook.com` to load the "Like" button; as a result, Doubleclick and Facebook learn about that user's visit to CNN. Cookie-based trackers re-identify users by setting unique identifiers in browser cookies, which are then automatically included with requests to the tracker's domain. Figure 2.1 shows a basic example; we discuss more complex cookie-based tracking behaviors in Section 2.4. Though cookie-based tracking is extremely common [147], other types of tracking behaviors have also emerged, including the use of other client-side storage mechanisms, such as HTML5 LocalStorage, or the use of browser and/or machine fingerprinting to re-identify users without the need to store local state [134, 43].

Because these embedded trackers are often invisible to users and not visited intentionally, there has been growing concern about the privacy implications of third-party tracking. In recent years, it has been the subject of repeated policy discussions (Mayer and Mitchell provide an overview as of 2012 [122]); simultaneously, the computer science research community has studied tracking mechanisms (e.g., [147, 134, 177, 122]), measured their preva-

lence (e.g., [147, 51, 102, 13]), and developed new defenses or privacy-preserving alternatives (e.g., [19, 164, 56, 64, 149]). We discuss related works further in Section 2.3.

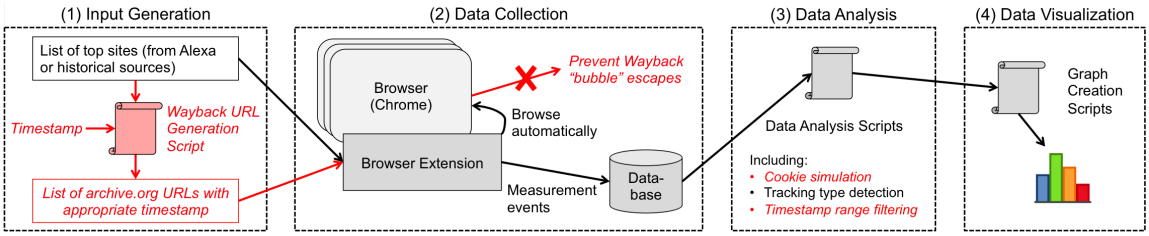



Figure 2.2: Overview of our infrastructure, TrackingExcavator, organized into four pipeline stages. Red/italic elements apply only to "Wayback mode" for historical measurements, while black/non-italics elements apply also to present-day measurements.

However, the research community's interest in web tracking is relatively recent, with the earliest measurements (to our knowledge) beginning in 2005 [102], and each study using a different methodology and measuring a different subset of known tracking techniques (see Englehardt et al. [49] for a comprehensive list of such studies). The practices of embedding third-party content and targeted advertising on websites predate these first studies [112], and longitudinal studies have been limited. However, longitudinal studies are critical to ensure the sustained effects of transparency [161] and to contextualize future measurements. Thus, to help ground technical and policy discussions surrounding web tracking in historical trends, we ask: how has the third-party tracking ecosystem evolved over the lifetime of the web?

We investigate questions such as:

- How have the **numbers, identities, and behaviors** of dominant trackers changed over time?
- How has the **scope** of the most popular trackers (i.e., the number of websites on which they are embedded) changed over time?
- How has the **prevalence** of tracking changed over time? For example, do websites include many more third-party trackers now than they did in the past?
- How have the **behaviors** of web trackers (e.g., JavaScript APIs used) changed over time?

By answering these questions, we are to able provide a systematic and longitudinal view of third-party web tracking over the last 20 years, retroactively filling this gap in the research literature, shedding a light on the evolution of third-party tracking practices on the web, and informing future technical and policy discussions.

**The Wayback Machine.** To conduct our archeological study, we rely on data from the Internet Archive's Wayback Machine (`https://archive.org`). Since 1996, the Wayback Machine has archived full webpages, including JavaScript, stylesheets, and any resources (including third-party JavaScript) that it can identify statically from the site contents. It mirrors past snapshots of these webpages on its own servers; visitors to the archive see the pages as they appeared in the past, make requests for all resources from the Wayback Machine's archived copy, and execute all JavaScript that was archived. We evaluate the completeness of the archive, particularly with respect to third-party requests, in Section 2.6.

### *2.3 Additional Related Work*

**Tracking and Defenses.** Third-party tracking has been studied extensively in recent years, particularly through analysis and measurements from 2005 to present [147, 100, 101, 72, 103, 63, 80, 82, 83, 102, 49, 50]. A few studies have considered mobile, rather than desktop, browser tracking [51, 67]. Beyond explicit stateful (e.g., cookie-based) tracking, recent work has studied the use of browser and machine fingerprinting techniques to re-identify and track users [43, 86, 177, 12, 134, 13]. Others have studied the possible results of tracking, including targeted ads [175, 106], personalized search [69], and price discrimination [169].

User-facing defenses against tracking range from browser extensions like Ghostery [58] and Privacy Badger [46] to research proposals (e.g. [68, 23]). Researchers have also designed privacy-preserving alternatives including privacy-preserving ads [64, 164, 56, 145], social media widgets [149, 39, 99], and analytics [19]. Others have studied user attitudes towards tracking and targeted advertising (e.g., [124, 107, 165]). Our study shows the increased prevalence of tracking over time, suggesting that designing and supporting these defenses for privacy-sensitive users is as important as ever.

**Wayback Machine and other Longitudinal Measurements.** Others have used the Wayback Machine for historical measurements to predict whether websites will become malicious [157] and to study JavaScript inclusion [136] and website accessibility [66]; to recover medical references [170]; to analyze social trends [84]; and as evidence in legal cases [47]. Others [130] found that websites are accurately reflected in the archive. These studies noted similar limitations as we did, as well as ways it has changed over time [88]. Finally, researchers have studied other aspects of the web and Internet longitudinally without the use of archives, including IPv6 adoption [34], search-engine poisoning [108], privacy notices [125], and botnets [171].

## *2.4 Measurement Infrastructure: TrackingExcavator*

To conduct a longitudinal study of web tracking using historical data from the Wayback Machine, we built a tool, TrackingExcavator, with the capability to (1) detect and analyze third-party tracking-related behaviors on a given web page, and (2) run that analysis over historical web pages archived and accessed by the Wayback Machine. In this section, we introduce TrackingExcavator. Figure 2.2 provides and overview of TrackingExcavator, which is organized into four pipeline stages:

**(1) Input Generation (Section 2.4.1):** TrackingExcavator takes as input a list of top-level sites on which to measure tracking behaviors (such as the Alexa top 500 sites), and, in "Wayback mode," a timestamp for the desired archival time to create `archive.org` URLs.

**(2) Data Collection (Section 2.4.2):** TrackingExcavator includes a Chrome browser extension that automatically visits the pages from the input set and collects tracking-relevant data, such as third-party requests, cookies, and the use of certain JavaScript APIs.

**(3) Data Analysis (Section 2.4.3):** TrackingExcavator processes collected measurement events to detect and categorize third-party web tracking behaviors.

**(4) Data Visualization:** Finally, we process our results into visual representations (in-

cluded in Section 2.5).

### *2.4.1 Input Generation*

In the input generation phase, we provide TrackingExcavator with a list of top-level sites to use for measurement. For historical measurements, TrackingExcavator must take a list of top-level URLs along with historical timestamps and transform them into appropriate URLs on `archive.org`. For example, the URL for the Wayback Machine's February 10, 2016 snapshot of `https://www.usenix.org/conference/usenixsecurity16` is `https://web.archive.org/web/20160210050636/https://www.usenix.org/conference/usenixsecurity16`.

We use the Memento API to find the nearest archived snapshot of a website occurring before the specified measurement date [85]. Though this process ensures a reasonable timestamp for the top-level page, embedded resources may have been archived at different times [18]. During analysis, we thus filter out archived resources whose timestamps are more than six months from our measurement timestamp, to ensure minimal overlap and sufficient spacing between measurements of different years.

### *2.4.2 Data Collection*

To collect data, TrackingExcavator uses a Chrome extension to automatically visit the set of input sites. Note that we cannot log into sites, since the Wayback Machine cannot act as the original server. Our browser is configured to allow third-party cookies as well as popups, and we visit the set of sites twice: once to prime the cache and the cookie store (to avoid artifacts of first-time browser use), and once for data collection. During these visits, we collect the following information relevant to third-party web tracking and store it in a local database:

- All request and response headers (including `set-cookie`).
- All cookies programmatically set by JavaScript (using `document.cookie`).
- All accesses to fingerprint-related JavaScript APIs, as described below.
- For each request: the requested URL, (if available) the referrer, and (if available)

information about the originating tab, frame, and window.

We later process this data in the analysis phase of TrackingExcavator's pipeline (Section 2.4.3 below).

**Fingerprint-Related APIs.** Since cookie-based web tracking is extremely common (i.e., it is "classic" web tracking), we focus largely on it — and third-party requests in general — to capture the broadest view of the web tracking ecosystem over time. However, we also collect information about the uses of other, more recently emerged tracking-related behaviors, such as JavaScript APIs that may be used to create browser or machine fingerprints [43, 134]. To capture any accesses a webpage makes to a fingerprint-related JavaScript API (such as `navigator.userAgent`), TrackingExcavator's Chrome extension Content Script overwrites these APIs on each webpage to (1) log the use of that API and (2) call the original, overwritten function. The set of APIs that we hook was collected from prior work on fingerprint-based tracking [43, 134, 137, 12, 13] and is provided in Appendix 2.7.

**Preventing Wayback "Escapes".** In archiving a page, the Wayback Machine transforms all embedded URLs to archived versions of those URLs (similar to our own process above). However, sometimes the Wayback Machine fails to properly identify and rewrite embedded URLs. As a result, when that archived page is loaded on `archive.org`, some requests may "escape" the archive and reference resources on the live web [88, 26]. In our data collection phase, we block such requests to the live web to avoid anachronistic side effects. However, we record the domain to which such a request was attempted, since the archived site did originally make that request, and thus we include it in our analysis.

### *2.4.3 Data Analysis*

In designing TrackingExcavator, we chose to separate data collection from data analysis, rather than detecting and measuring tracking behaviors on the fly. This modular architecture simplifies data collection and isolates it from possible bugs or changes in the analysis pipeline — allowing us to rerun different analyses on previously collected data (e.g., to

retroactively omit certain domains).

**"Replaying" Events.** Our analysis metaphorically "replays" collected events to simulate loading each page in the measurement. For historical measurements, we modify request headers to replace "live web" `Set-Cookie` headers with `X-Archive-Orig-Set-Cookie` headers added by `archive.org`, stripping the Wayback Machine prefixes from request and referrer URLs, and filling our simulated cookie jar (described further below). During the replay, TrackingExcavator analyzes each event for tracking behaviors.

**Classifying Tracking Behaviors.** For *cookie-based trackers*, we base our analysis on a previously published taxonomy [147].[2] We summarize — and augment — that taxonomy here. Note that a tracker may fall into multiple categories, and that a single tracker may exhibit different behaviors across different sites or page loads:

1. *Analytics Tracking*: The tracker provides a script that implements website analytics functionality. Analytics trackers are characterized by a script, sourced from a third party but run in the first-party context, that sets first-party cookies and later leaks those cookies to the third-party domain.
2. *Vanilla Tracking*: The tracker is included as a third party (e.g., an iframe) in the top-level page and uses third-party cookies to track users across sites.
3. *Forced Tracking*: The tracker forces users to visit its domain directly — for example, by opening a popup or redirecting the user to a full-page ad — allowing it to set cookies from a first-party position.
4. *Referred Tracking*: The tracker relies on another tracker to leak unique identifiers to it, rather than on its own cookies. In a hypothetical example, `adnetwork.com` might set its own cookie, and then explicitly leak that cookie in requests to referred tracker `ads.com`. In this case, `ads.com` need not set its own cookies to perform tracking.
5. *Personal Tracking*: The tracker behaves like a Vanilla tracker but is visited by the user directly in other contexts. Personal trackers commonly appear as social widgets (e.g.,

[2]We are not aware of other taxonomies of this granularity for cookie-based tracking.

"Like" or "tweet" buttons).

In addition to these categories previously introduced [147], we discovered an additional type of tracker related to but subtly different from Analytics tracking:

6. *Referred Analytics Tracking:* Similar to an Analytics tracker, but the domain which sets a first-party cookie is *different* from the domain to which the first-party cookie is later leaked.

Beyond cookie-based tracking behaviors, we also consider the use of fingerprint-related JavaScript APIs, as described above. Though the use of these APIs does not necessarily imply that the caller is fingerprinting the user — we know of no published heuristic for determining fingerprinting automatically — but the use of many such APIs may suggest *fingerprint-based tracking*.

Finally, in our measurements we also consider *third-party requests* that are not otherwise classified as trackers. If contacted by multiple domains, these third-parties have the *ability* to track users across sites, but may or may not actually do so. In other words, the set of all domains to which we observe a third-party request provides an upper bound on the set of third-party trackers.

We tested TrackingExcavator's detection and classification algorithms using a set of test websites that we constructed and archived using the Wayback Machine, triggering each of these tracking behaviors.

**Reconstructing Archived Cookies.** For many tracking types, the presence or absence of cookies is a key factor in determining whether the request represents a tracking behavior. In our live measurements, we have the actual `Cookie` headers attached by Chrome during the crawl. On archived pages, the Wayback Machine includes past `Set-Cookie` headers as `X-Archive-Orig-Set-Cookie` headers on archived responses. To capture the cookies that would have actually been set during a live visit to that archived page, TrackingExcavator must simulate a browser cookie store based on these archival cookie headers and JavaScript cookie set events recorded during data collection.

Unfortunately, cookie engines are complicated and standards non-compliant in major browsers, including Chrome [31]. Python's cookie storage implementation is compliant with RFC 2965, obsoleted by RFC 6265, but these standards proposals do not accurately represent modern browser practices [55, 35, 22]. For efficiency, we nevertheless use Python's cookie jar rather than attempting to re-implement Chrome's cookie engine ourselves.

We found that Python's cookie jar computed cookies exactly matching Chrome's for only 71% of requests seen in a live run of the top 100. However, for most types of tracking, we only need to know whether *any* cookies would have been set for the request, which we correctly determine 96% of the time. Thus our tool captures most tracking despite using Python's cookie jar.

**Classifying Personal Trackers in Measurements.** For most tracker types, classification is independent of user behaviors. Personal trackers, however, are distinguished from Vanilla trackers based on whether the user visits that domain as a top-level page (e.g., Facebook or Google). To identify likely Personal trackers in automated measurement, we thus develop a heuristic for user browsing behaviors: we use popular sites from each year, as these are (by definition) sites that many users visited.

Alexa's top sites include several that users would not typically visit directly, e.g., `googleadservices.com`. Thus, we manually examined lists of popular sites for each year to distinguish between domains that users *typically* visit intentionally (e.g., Facebook, Amazon) from those which ordinary users never or rarely visit intentionally (e.g., ad networks or CDNs). Two researchers independently classified the domains on the Alexa top 100 sites for each year where we have Alexa data, gathering information about sites for which they were unsure. The researchers examined 435 total domains: for the top 100 domains in 2016, they agreed on 100% and identified 94 sites as potential Personal trackers; for the 335 additional domains in the previous years' lists, they agreed on 95.4% and identified 296 Personal tracker domains.

## 2.5 Historical Web Tracking Measurements

We now turn to our longitudinal study of third-party cookie-based web tracking from 1996-2016.

**Datasets.** We focus our investigation on the most popular websites each year, for two reasons: first, trackers on these sites are (or were) able to collect information about the greatest number of users; second, popular sites are crawled more frequently by the Wayback Machine (if permitted by `robots.txt`). We thus need historical lists of the top sites globally on the web.

*2003-2016: Alexa.* For 2010-2016, we use Wayback Machine archives of Alexa's top million sites list (a `csv` file). For 2003-2009, we approximate the top 500 by scraping Alexa's own historical API (when available) and archives of individual Alexa top 100 pages. Because of inconsistencies in those sources, our final lists contain 459-500 top sites for those years.

*1996-2002: Popular Links from Homepages.* In 2002, only the Alexa top 100 are available; before 2002, we only have ComScore's list of 20 top sites [173]. Thus, to build a list of 500 popular sites for the years 1996-2002, we took advantage of the standard practice at the time of publishing links to popular domains on personal websites. Specifically, we located archives of the People pages of the Computer Science or similar department at the top 10 U.S. CS research universities as of 1999, as reported in that year by U.S. News Online [2]. We identified the top 500 domains linked to from the homepages accessible from those People pages, and added any ComScore domains that were not found by this process. We ran this process using People pages archived in 1996 and 1999; these personal pages were not updated or archived frequently enough to get finer granularity. We used the 1996 list as input to our 1996, 1997 and 1998 measurements, and the 1999 list as input for 1999-2002.

### 2.5.1 Prevalence of Tracking Behaviors over Time

We begin by studying the prevalence of tracking behaviors over time: how many unique trackers do we observe, what types of tracking behaviors do those trackers exhibit, and how

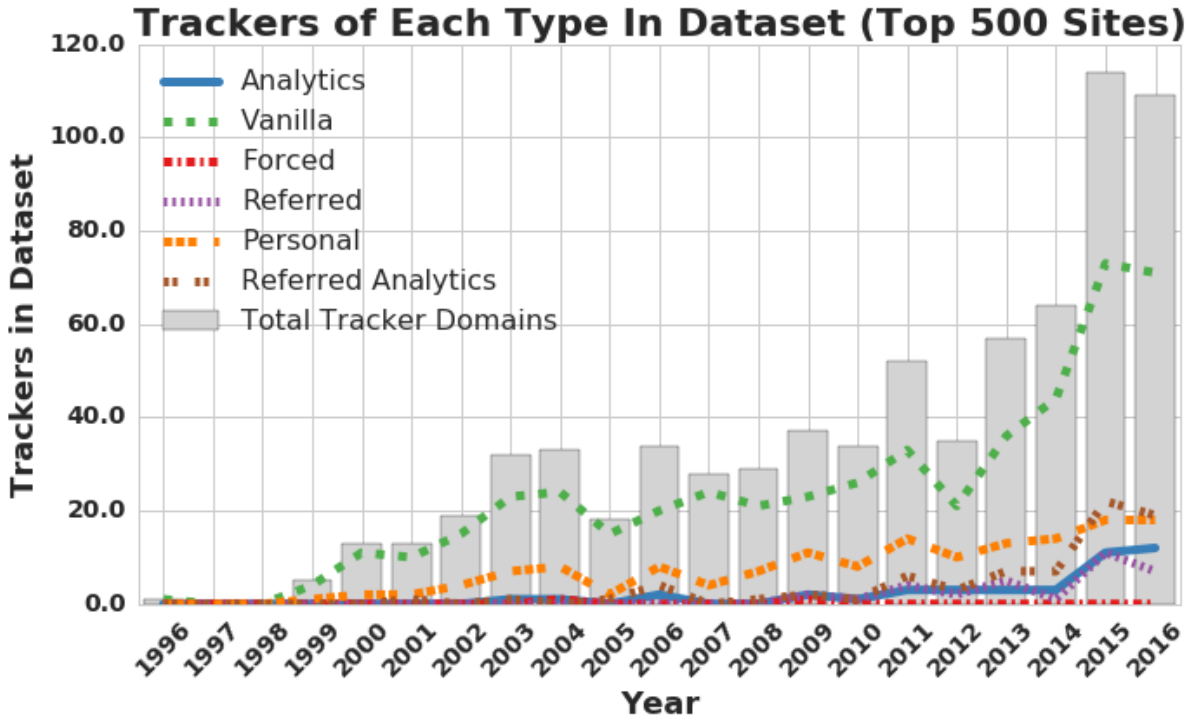



Figure 2.3: Evolution of tracker types over time. The grey bars show the total number of tracking domains present in each dataset, and the colored lines show the numbers of trackers with each type of tracking behavior. A single tracker may have more than one behavior in the dataset (e.g., both Vanilla and Analytics), so the sum of the lines might be greater than the bar.

many trackers appear on sites over time?

**Prevalence and Behaviors of Unique Trackers.** Figure 2.3 shows the total number of unique trackers observed over time (the grey bars) and the prevalence of different tracking behavior types (the lines) for the top 500 sites from 1996-2016. Note that trackers may exhibit more than one behavior across sites or on a single site, so the sum of the lines may be greater than the height of the bar. We note that the particularly large bars in 2015 and 2016 may reflect not only a change in tracking prevalence but also changes in the way the Wayback Machine archived the web. See Table 2.5 for validation against live data which suggest that actual growth may have been smaller and more linear, similar to past years.

| Year | 1Type | 2Type | 3Type | 4Type |
|---|---|---|---|---|
| 1996 | 100.00% (1) | 0 | 0 | 0 |
| 1998 | 0 | 0 | 0 | 0 |
| 2000 | 100.00% (13) | 0 | 0 | 0 |
| 2002 | 100.00% (19) | 0 | 0 | 0 |
| 2004 | 96.97% (32) | 3.03% (1) | 0 | 0 |
| 2006 | 100.00% (34) | 0 | 0 | 0 |
| 2008 | 100.00% (29) | 0 | 0 | 0 |
| 2010 | 94.12% (32) | 2.94% (1) | 2.94% (1) | 0 |
| 2012 | 88.57% (31) | 11.43% (4) | 0 | 0 |
| 2014 | 93.75% (60) | 4.69% (3) | 1.56% (1) | 0 |
| 2016 | 86.24% (94) | 11.01% (12) | 2.75% (3) | 0 |

Table 2.1: Complexity of trackers, in terms of the percentage (and number) of trackers displaying one or more types of tracking behaviors across the top 500 sites.

We make several observations. First, we see the emergence of different tracking behaviors: the first cookie-based tracker in our data is from 1996: `microsoft.com` as a Vanilla tracker on `digital.net`. The first Personal tracker to appear in our dataset is in 1999: `go.com` shows up on 5 different sites that year, all also owned by Disney: `disney.com, espn.com, sportszone.com, wbs.net,` and `infoseek.com` (acquired by Disney mid-1999 [1], before the date of our measurement). The existence of a Personal tracker that only appeared on sites owned by the same company differs from today's Personal tracking ecosystem, in which social media widgets like the Facebook "Like" button appear on many popular sites unaffiliated with that tracker (Facebook, in this case) [147].

More generally, we see a marked increase in quantities of trackers over time, with rises in all types of tracking behavior. One exception is Forced trackers — those relying on popups — which are rare and peaked in the early 2000s before popup blockers became default (e.g., in 2004 for Internet Explorer [132]). Indeed, we see third-party popups peak significantly in 2003 and 2004 (17 and 30 popups, respectively, compared to an annual mean of about 4), though we could not confirm all as trackers for Figure 2.3. Additionally, we see an increasing variety of tracking behavior over time, with early trackers nearly all simply Vanilla, but more recent rises in Personal, Analytics, and Referred tracking.

We can also consider the complexity of individual trackers, i.e., how many distinct track-

| Year | Most Prolific API-user | Num APIs Used | Coverage |
|---|---|---|---|
| 1998 | realhollywood.com | 2 | 1 |
| 1999 | go2net.com | 2 | 1 |
| 2000 | go.com | 6 | 2 |
| 2001 | akamai.net | 8 | 15 |
| 2002 | go.com | 10 | 2 |
| 2003 | bcentral.com | 5 | 1 |
| 2004 | 163.com | 9 | 3 |
| 2005 | 163.com | 8 | 1 |
| 2006 | sina.com.cn | 11 | 2 |
| 2007 | googlesyndication.com | 8 | 24 |
| 2008 | go.com | 12 | 1 |
| 2009 | clicksor.com | 10 | 2 |
| 2010 | tribalfusion.com | 17 | 1 |
| 2011 | tribalfusion.com | 17 | 2 |
| 2012 | imedia.cz | 12 | 1 |
| 2013 | imedia.cz | 13 | 1 |
| 2014 | imedia.cz | 13 | 1 |
| 2015 | aolcdn.com | 25 | 5 |
| 2016 | aolcdn.com | 25 | 3 |

Table 2.2: Most prolific API-users, with ties broken by coverage (number of sites on which they appear) for each year. The maximum number of APIs used increases over time, but the max API users are not necessarily the most popular trackers.

ing behaviors they exhibit over each year's dataset. (Note that some behaviors are exclusive, e.g., a tracker cannot be both Personal and Vanilla, but others are nonexclusive.) Table 2.1 suggests that there has been some increase in complexity in recent years, with more trackers exhibiting two or even three behaviors. Much of this increase is due to the rise in Referred or Referred Analytics trackers, which receive cookie values shared explicitly by other trackers in addition to using their own cookies in Vanilla behavior.

**Fingerprint-Related APIs.** We measured the use of Javascript APIs which can be used to fingerprint browsers and persist identifiers even across cookie deletion. Though the use of these APIs does not necessarily imply that they are used for tracking (and we know of no published heuristic for correlating API use with genuine fingerprinting behaviors), the use of these APIs nevertheless allows third parties to gather potentially rich information about users and their machines. The full list of 37 fingerprint-related APIs we measure (based on prior work [43, 134, 137, 12, 13]) is in Appendix 2.7.

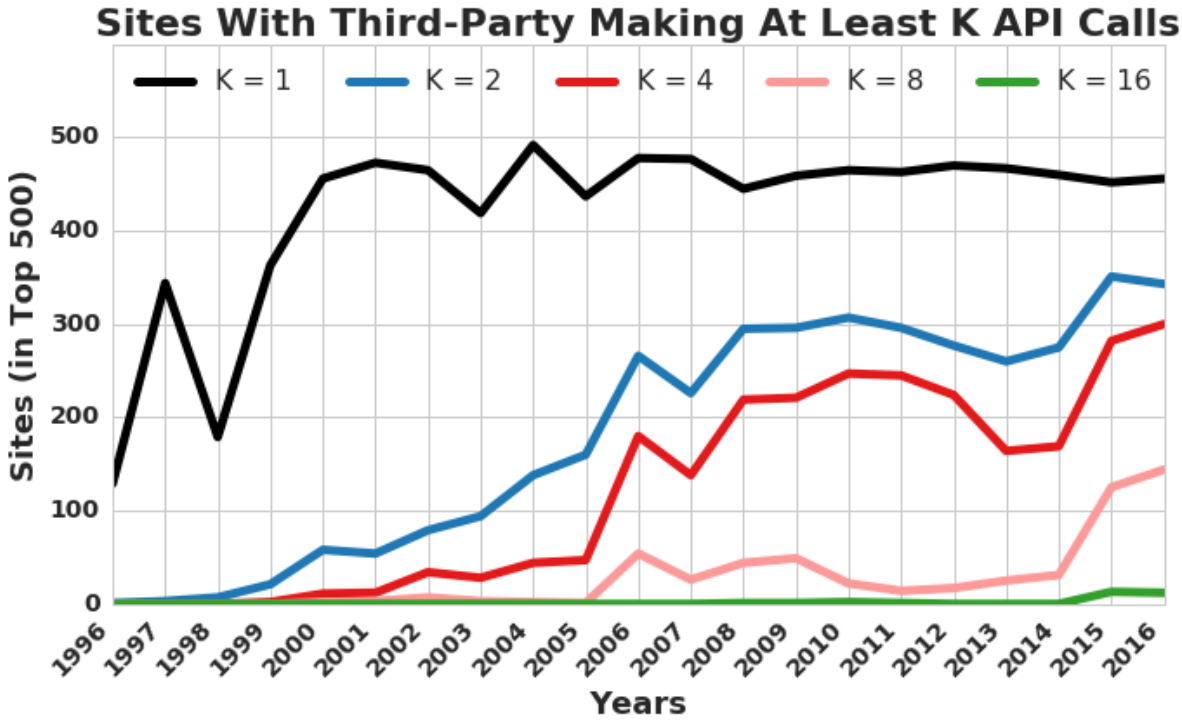



Figure 2.4: Number of sites in each year with a tracker that calls (on that site) at least K (of our 37) fingerprint-related APIs.

We now consider third parties that are prolific users of fingerprint-related APIs, calling many APIs on each site. Table 2.2 shows the tracker in each year that calls the most APIs on a single site. Ties are broken by the choosing the third party that appears on the largest number of sites. Maximum usage of APIs has increased over time, but we observe that the most prolific API users are not the most popular cookie-based trackers. Although we only identify API uses within JavaScript, and not how their results are used, we note that increasing use of these APIs implies increased power to fingerprint, especially when combined with non-Javascript signals such as HTTP headers and plugin behavior. For example, Panopticlick derived 18 bits of entropy about remote browsers from a subset of these APIs plus HTTP headers and information from plugins [43].

Beyond the power of the most prolific fingerprint-related API users growing, we also find that more sites include more trackers using these APIs over time. Figure 2.4 shows the

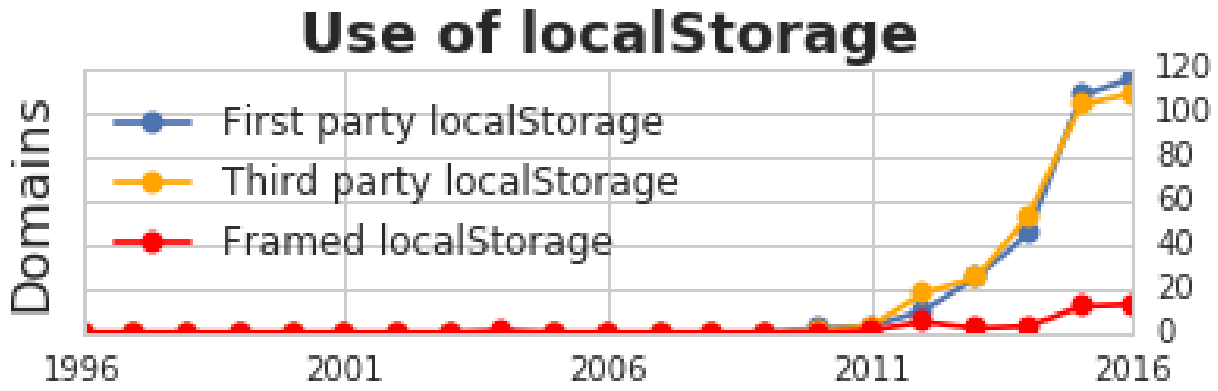



Figure 2.5: Domains using `window.localStorage`. First party usages are uses in the top frame of a web page by a script loaded from the web page's own domain. Third party usages are those also in the top frame of a page but by a script loaded from a third party. Framed uses are those inside of an iframe.

number of sites in each year containing a tracker that calls, on that site, at least $K$ of the 37 fingerprinting APIs. Although many sites contain and have contained trackers that use at least 1 API (typically `navigator.userAgent`, common in browser compatibility checks), the number of sites containing trackers that call 2 or more APIs has risen significantly over time.

In addition to fingerprint-related APIs, we also examine the use of HTML5 LocalStorage, a per-site persistent storage mechanism standardized in 2009 in addition to cookies. Figure 2.5 shows that the use of the `localStorage` API rises rapidly since its introduction in 2009, indicating that tracking defenses should increasingly consider on storage mechanisms beyond cookies.

**Third Parties Contacted.** We now turn our attention to the number of third parties that users encounter as they browse the web. Even third parties not confirmed as trackers have the *potential* to track users across the web, and as we discovered in Section 2.6, many third parties in archived data may in fact be confirmed trackers for which the Wayback Machine simply archived insufficient information. Figure 2.6 thus shows the distributions of how many third parties the top 500 sites contacted in each year. We see a rise in the median

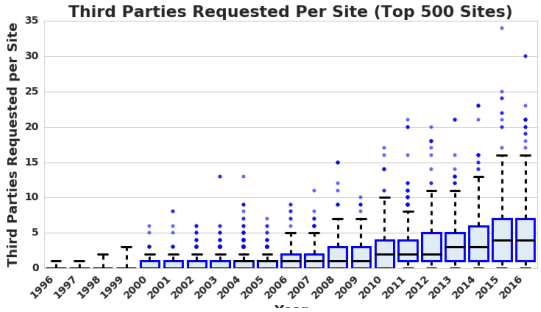



Figure 2.6: Distributions of third-party requests for the top 500 sites 1996-2016. Center box lines are medians, whiskers end at 1.5*IQR. The increase in both medians and distributions of the data show that more third-parties are being contacted by popular sites in both the common and extreme cases.

number of third parties contacted — in other words, more sites are giving more third parties the opportunity to track users.

Figure 2.7 provides a different view of similar data, showing the distribution of the top sites for each year by number of distinct third parties contacted. In the early 2000s, only about 5% of sites contacted at least 5 third parties, while in 2016 nearly 40% of sites did so. We see a maximum in 2015, when one site contacted 34 separate third-parties (a raw number that is likely underestimated by the Wayback Machine's data)!

### *2.5.2 Top Trackers over Time*

We now turn to an investigation of the top trackers each year: who are the top players in the ecosystem, and how wide is their view of users' browsing behaviors?

**Coverage of Top Trackers.** We define the *coverage* of a set of trackers as the percentage of total sites from the dataset for which at least one of those trackers appears. For a single tracker, its coverage is the percentage of sites on which it appears. Intuitively, coverage suggests the concentration of tracking ability — greater coverage allows trackers to build larger browsing profiles. This metric reaches toward the core privacy concern of tracking, that certain entities may know nearly everything a person does on the web. We consider trackers by domain name, even though some trackers are in fact owned by the same company

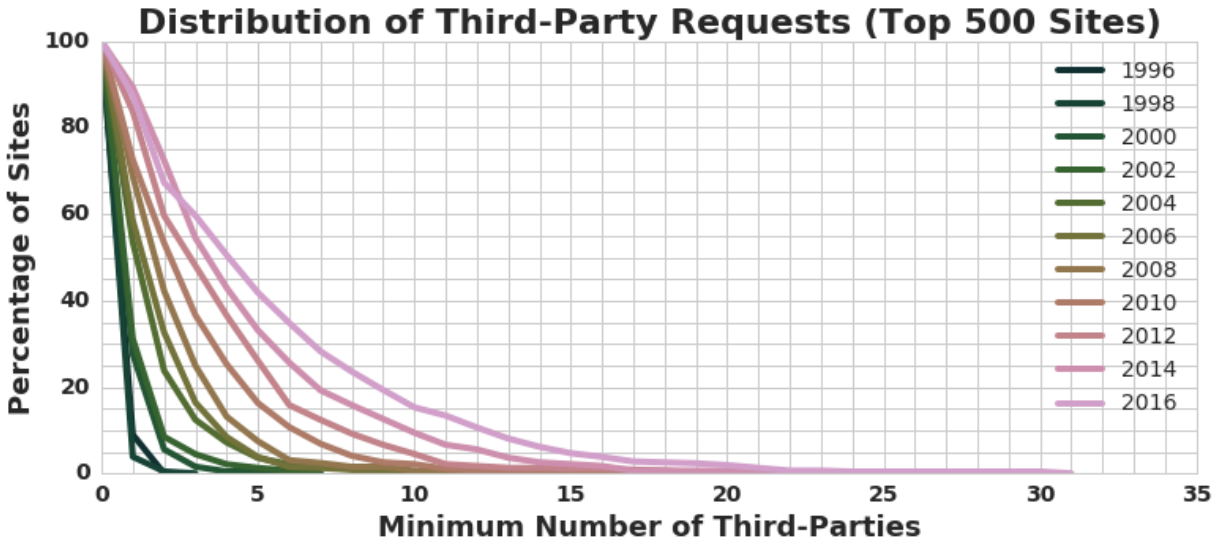



Figure 2.7: Distribution of top sites for each year by number of unique third-parties (tracking-capable domains) they contact. In later years, more sites appear to contact more third parties.

(e.g., Google owns `google-analytics.com`, `doubleclick.net`, and the "+1" button served from `google.com`), because a business relationship does not imply that the entities share data, though some trackers may indeed share information out of public view.

Figure 2.8 illustrates the growth of tracker coverage over time. It considers both the single domain with the highest coverage for each year (Top 1 Tracker) as well as the combined coverage of the union of the top 5, 10 and 20 trackers. Confirming the lesson from Section 2.6.2, the coverage rates we see for third party domains in the archive are similar to live coverage of *confirmed* Vanilla cookie-based trackers.

Clearly, the coverage of top trackers has risen over time, suggesting that a small number of third parties can observe an increasing portion of user browsing histories.

**Popular Trackers over Time.** Who are these top trackers? Figure 2.9 shows the rise and fall of the top two trackers ("champions") for each year. To create this figure, we make use of the lesson in Section 2.6.4 to manually label known popular confirmed trackers. We identified the two domains with the highest third-party request coverage for each year, omitting cases

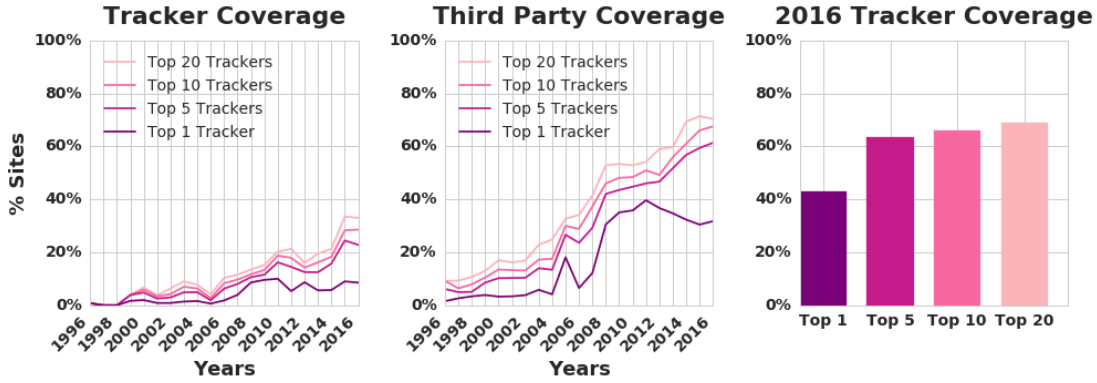



Figure 2.8: The growth in the coverage (percentage of top 500 sites tracked) of the top 1/5/10/20 trackers for each year is shown in the first and second panels, for all confirmed trackers and for all third parties respectively. The right hand panel shows the values on the live web for confirmed trackers, with the top 5 trackers covering about 70% of all sites in the dataset. Note that top third party coverage in the archive is an excellent proxy for modern confirmed tracker coverage today.

where the most popular tracker in a year appeared on only one site. We manually verified that 12/19 of these domains were in fact trackers by researching the domain, owning company, archived behavior and context, and modern behaviors (if applicable). Based on this analysis, we are able to assess the change in tracking behaviors even of domains for whom cookies are lost in the archive (e.g., `doubleclick.net`). In particular, this analysis reveals trends in the trackers with the most power to capture profiles of user behavior across many sites.

We find that in the early 2000s, no single tracker was present on more than 10% of top sites, but in recent years, `google-analytics.com` has been present on nearly a third of top sites and 2-4 others have been present on more than 10% and growing. Some, such as `doubleclick.net` (acquired by Google in 2008) have been popular throughout the entire time period of the graph, while others, such as `scorecardresearch.com`, have seen a much more recent rise.

We note that `google-analytics.com` is a remarkable outlier with nearly 35% coverage in 2011. Google Analytics is also an outlier in that it is one of only two non-cross-site trackers among the champions (`gstatic.com`, a Referred Analytics tracker, is the other). As an Analytics type tracker, Google Analytics trackers users only within a single site, meaning

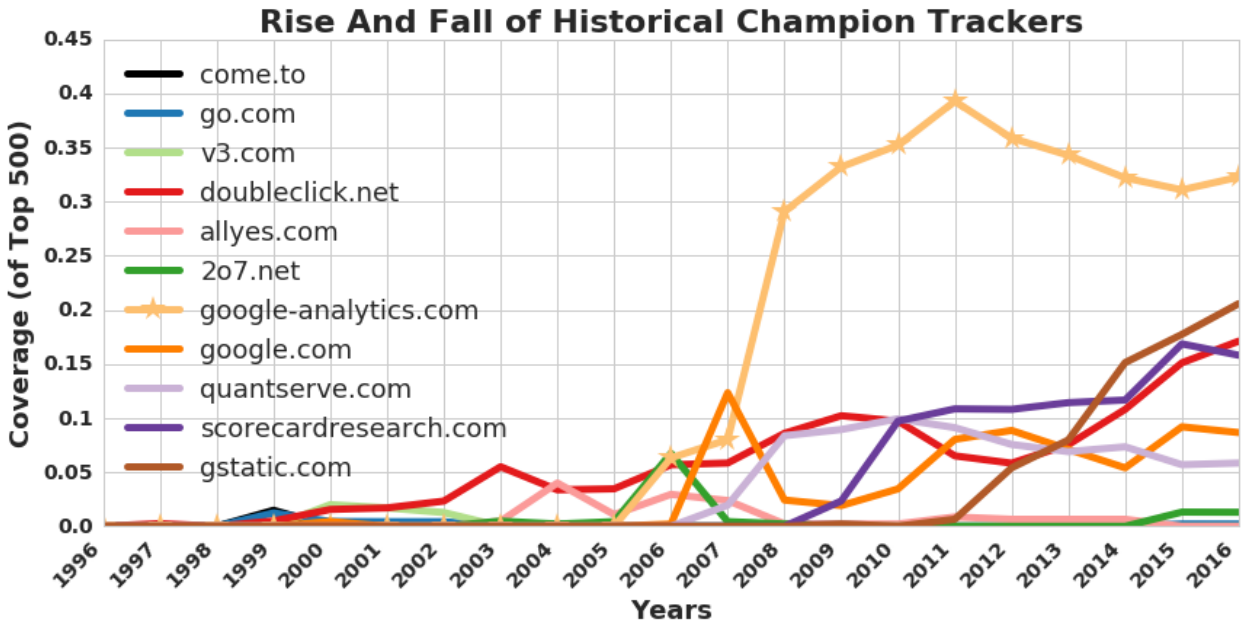



Figure 2.9: This figure depicts variations in site coverage for a number of the most popular confirmed trackers from years across the studied period. We call the two trackers embedded on the most sites in a given year the "champions" of that year, filtered by manual classification as described in the text.

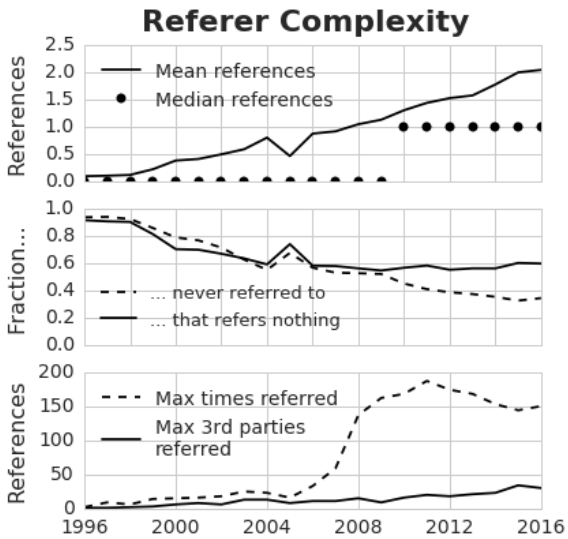



Figure 2.10: Changes in the frequency with which domains are referred to or refer to other domains (based on HTTP Referer).

that its "coverage" is arguably less meaningful than that of a cross-site tracker. However, we observe that Google Analytics *could* track users across sites via fingerprinting or by changing its behavior to store tracking cookies. This observation highlights the need for repeated measurements studies that provide transparency on the web: with a simple change to its tracking infrastructure, Google Analytics could begin to track users across 40% of the most popular sites on the web overnight. Thus, Google's decision not to structure Google Analytics in this way has a tremendous impact on user privacy.

### *2.5.3 Evolution of the Tracking Ecosystem*

Finally, we consider the tracking ecosystem as a whole, focusing on relationships between different trackers. We find a remarkable increase in the complexity of these relationship over time. Again we consider only relationships observable using TrackingExcavator, not external information about business relationships.

To study these relationships, we construct the graph of referring relationships between elements on pages. For example, if we observe a third-party request from `example.com` to

`tracker.com`, or from `tracker.com` referring to `tracker2.com`, the nodes for those domains in the graph will be connected by edges.

We find a significant increase in complexity over time by examining several properties of this graph (Figure 2.10). Over time, the mean number of referrals outward from domains increases (top of Figure 2.10), while the number of domains that are never referred to by other domains or never refer outward steadily decreases (middle of Figure 2.10). Meanwhile, the maximum number of domains that refer to a single domain increases dramatically, suggesting that individual third parties in the web ecosystem have gradually gained increasing prominence and coverage. This reflects and confirms trends shown by other aspects of our data (Figures 2.9 and 2.8). These trends illuminate an ecosystem of generally increasingly connected relationships and players growing in size and influence. Appendix 2.8 shows this evolution in graph form; the increase in complexity over time is quite striking.

### *2.5.4 Summary and Discussion*

We have uncovered trends suggesting that tracking has become more prevalent and complex in the 20 years since 1996: there are now more unique trackers exhibiting more types of behaviors; websites contact increasing numbers of third parties, giving them the opportunity to track users; the scope of top trackers has increased, providing them with a broader view of user browsing behaviors; and the complexity and interconnectedness of the tracking ecosystem has increased markedly.

From a privacy perspective, our findings show that over time, more third parties are in a position to gather and utilize increasing amounts of information about users and their browsing behaviors. This increase comes despite recent academic, policy, and media attention on these privacy concerns and suggests that these discussions are far from resolved. As researchers continue to conduct longitudinal measurements of web tracking going forward, our work provides the necessary historical context in which to situate future developments.

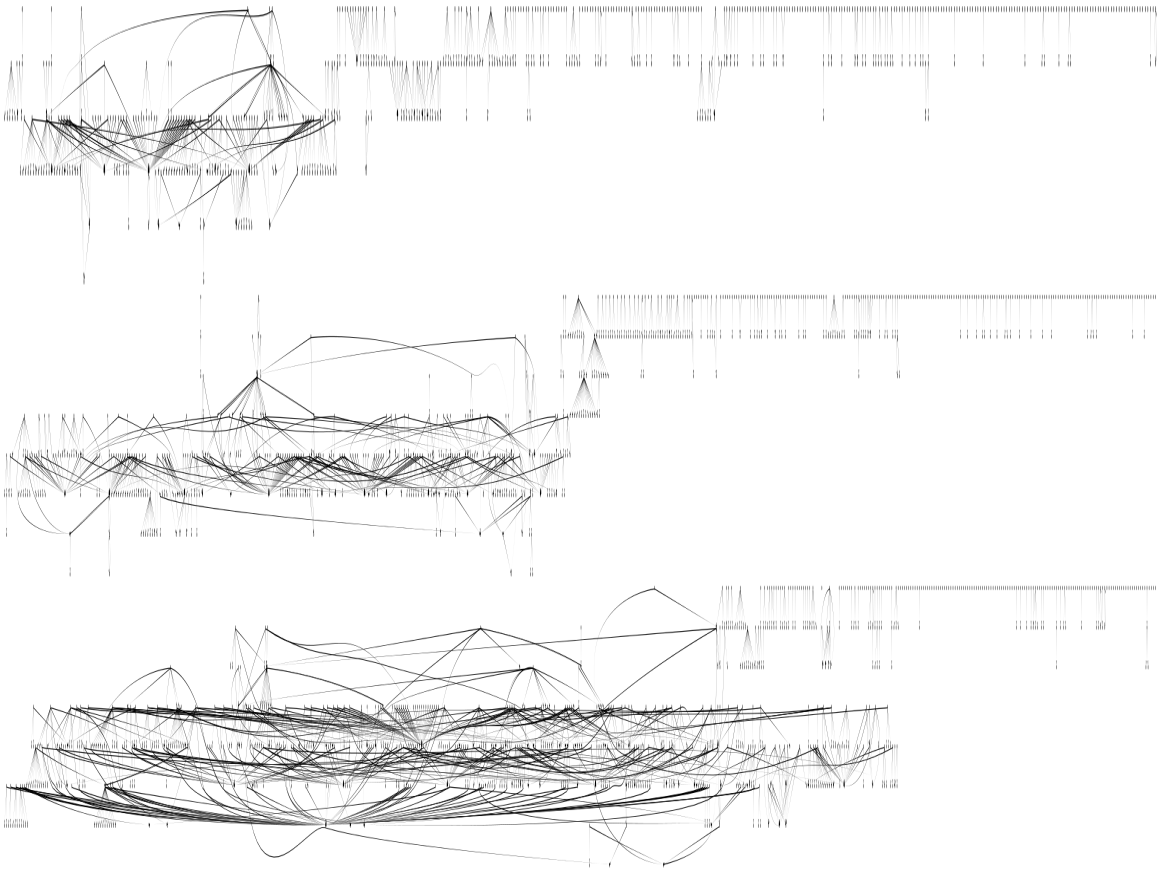


Figure 2.11: 2003, 2005 and 2007 referer graphs for the top 500, as seen in the Wayback Machine's archive. An from a domain `referer.com` to another domain `referred.com` is included if any URL from `referer.com` is seen to be the referer for any URL from `referred.com`. Note the increasing complexity of the graph over time. Note as well increased connectivity, with a larger percentage of the graph joined in a single component.

## *2.6 Evaluating the Wayback Machine as an Archaeological Data Source for Tracking*

The Wayback Machine provides a unique and comprehensive source of historical web data. However, it was not created for the purpose of studying third-party web tracking and is thus imperfect for that use. Nevertheless, the only way to study web tracking *prior* to explicit measurements targeting it is to leverage materials previously archived for other purposes. Therefore, before using the Wayback Machine's archived data, it is essential to systematically characterize and analyze its capabilities and flaws in the context of third-party tracking.

In this section we thus study the extent to which data from the Wayback Machine allows us to study historical web tracking behaviors. Beyond providing confidence in the trends of web tracking over time that we present in Section 2.5, we view this evaluation of the Wayback Machine as a contribution of this chapter. While others have studied the quality of the Wayback Machine's archive, particularly with respect to the quality of the archived content displayed on the top-level page (e.g., [130, 88, 28]), we are the first to systematically study the quality of the Wayback Machine's data about *third-party requests*, the key component of web tracking.

To conduct our evaluation, we leverage four ground truth data sets collected from the live web in 2011, 2013, 2015, and 2016. The 2011 data was originally used in [147] and provided to us by those authors. All datasets contain classifications of third-party cookie-based trackers (according to the above taxonomy) appearing on the Alexa top 500 sites (from the time of each measurement). The 2015 and 2016 data was collected by TrackingExcavator and further contains all HTTP requests, including those not classified as tracking.[3] We plan to release our ground truth datasets from 2013, 2015, and 2016.

We organize this section around a set of lessons that we draw from this evaluation. We apply these lessons in our measurements in Section 2.5. We believe our findings can assist

[3]For comparison, the published results based on the 2011 dataset [147] measured tracking on the homepages of the top 500 websites as well as four additional pages on that domain; for the purposes of our work, we re-analyzed the 2011 data using only homepages.

| | August 1 | August 25 | September 1 |
|---|---|---|---|
| All Third-Parties | 324 | 304 | 301 |
| Analytics | 7 | 13 | 11 |
| Vanilla | 127 | 115 | 108 |
| Forced | 0 | 0 | 0 |
| Referred | 3 | 3 | 3 |
| Personal | 23 | 21 | 21 |
| Referred Analytics | 21 | 17 | 18 |

Table 2.3: Natural variability in the trackers observed on different visits to the Alexa top 100 in 2015. This variability can result from non-static webpage content, e.g., ad auctions that result in different winners.

future researchers seeking to use the Wayback Machine as a resource for studying tracking (or other web properties relying on third-party requests) over time.

### *2.6.1 Lesson (Challenge): The Wayback Machine provides a partial view of third-party requests*

A key question for using the Wayback Machine for historical measurements is: how complete is the archive's view of the past, both for the top-level pages and for the embedded content? In this lesson, we explore why its view is incomplete, surfacing challenges that we will overcome in subsequent lessons. We identify several reasons for the differences between the live and Wayback measurements, and quantify the effects of each.

**Variation Between Visits.** Different trackers and other third parties may appear on a site when it is loaded a second time, even if these views are close together; an example of this variation would be disparity in tracking behaviors between ads in an ad network.

To estimate the degree of variation between page views, we compare three live runs from August-September 2015 of the Alexa top 100 sites (Table 2.3). We find that variation between runs even a week apart is notable (though not enough to account for all of the differences between Wayback and live datasets). For the number of Vanilla trackers found, the August 25th and September 1st runs vary by 7 trackers, or 6%.

**Non-Archived and Blocked Requests.** There are several reasons that the Wayback

| Type of Blocking | | Fraction Missed |
|---|---|---|
| **Robots Exclusions** | Requests | 1115 / 56,173 (2.0%) |
| | URLs | 609 / 27,532 (2.2%) |
| | Domains | 18 / 1150 (1.6%) |
| **Not Archived** | Requests | 809 / 56,173 (1.4%) |
| | URLs | 579 / 27,532 (2.1%) |
| | Domains | 8 / 1150 (0.7%) |
| **Wayback Escapes** | Requests | 9025 / 56,173 (16.1%) |
| | URLs | 4730 / 27,532 (17.2%) |
| | Domains | 132 / 1150 (11.5%) |
| **Inconsistent Timestamps** | Requests | 404 / 56,173 (0.7%) |
| | URLs | 156 / 27,532 (0.6%) |
| | Domains | 55 / 1150 (4.8%) |

Table 2.4: For the archived versions of the Alexa top 500 sites from 2016, the fraction of requests, unique URLs, and unique domains affected by robots exclusion (403 errors), not archived (404), Wayback escapes (blocked by TrackingExcavator), or inconsistent timestamps (filtered by TrackingExcavator).

Machine may fail to archive a response to a request, or provide a response that TrackingExcavator must ignore (e.g., from a far different time than the one we requested or from the live web). We describe these conditions here, and evaluate them in the context of a Wayback Machine crawl of the top 500 pages archived in 2016, according to the 2016 Alexa top 500 rankings; we elaborate on this dataset in Section 2.5. Table 2.4 summarizes how often the various conditions occur in this dataset, for requests, unique URLs, and unique domains. In the case of domains, we count only those domains for which *all* requests are affected, since those are the cases where we will *never* see a cookie or any other subsequent tracking indicators for that domain.

*Robots.txt Exclusions (403 errors).* If a domain's `robots.txt` asks that it not be crawled, the Wayback Machine will respect that restriction and thus not archive the response. As a result, we will not receive any information about that site (including cookies, or use of Javascript) nor will we see any subsequent resources that would have resulted from that request.

We find that only a small fraction of all requests, unique URLs, and (complete) domains are affected by robots exclusion (Table 2.4). We note that robots exclusions are particularly

common for popular trackers. Of the 20 most popular trackers on the 2016 live dataset, 12 (60%) are blocked at least once by robots.txt in the 2016 Wayback measurement. By contrast, this is true for only 74/456, or 16.23%, of all Vanilla trackers seen in live.

*Other Failures to Archive (404 errors).* The Wayback Machine may fail to archive resources for any number of reasons. For example, the domain serving a certain resource may have been unavailable at the time of the archive, or changes in the Wayback Machine's crawler may result in different archiving behaviors over time. As shown in Table 2.4, missing archives are rare.

*URL Rewriting Failures (Wayback "Escapes").* Though the Wayback Machine's archived pages execute the corresponding archived JavaScript within the browser when TrackingExcavator visits them, the Wayback Machine does not execute JavaScript during its archival crawls of the web. Instead, it attempts to statically extract URLs from HTML and JavaScript to find additional sites to archive. It then modifies the archived JavaScript, rewriting the URLs in the included script to point to the archived copy of the resource. This process may fail, particularly for dynamically generated URLs. As a result, when TrackingExcavator visits archived pages, dynamically generated URLs not properly redirected to their archived versions will cause the page to attempt to make a request to the live web, i.e., "escape" the archive. TrackingExcavator blocks such escapes (see Section 2.4). As a result, the script never runs on the archived site, never sets a cookie or leaks it, and thus TrackingExcavator does not witness the associated tracking behavior.

We find that Wayback "escapes" are more common than robots exclusion or missing archives (Table 2.4): 16.1% of all requests attempted to "escape" (i.e., were not properly rewritten by the Wayback Machine) and were blocked by TrackingExcavator.

*Inconsistent Timestamps.* As others have documented [28], embedded resources in a webpage archived by the Wayback Machine may occasionally have a timestamp far from the timestamp of the top-level page. As described in Section 2.4, we ignore responses to requests for resources with timestamps more than six months away.

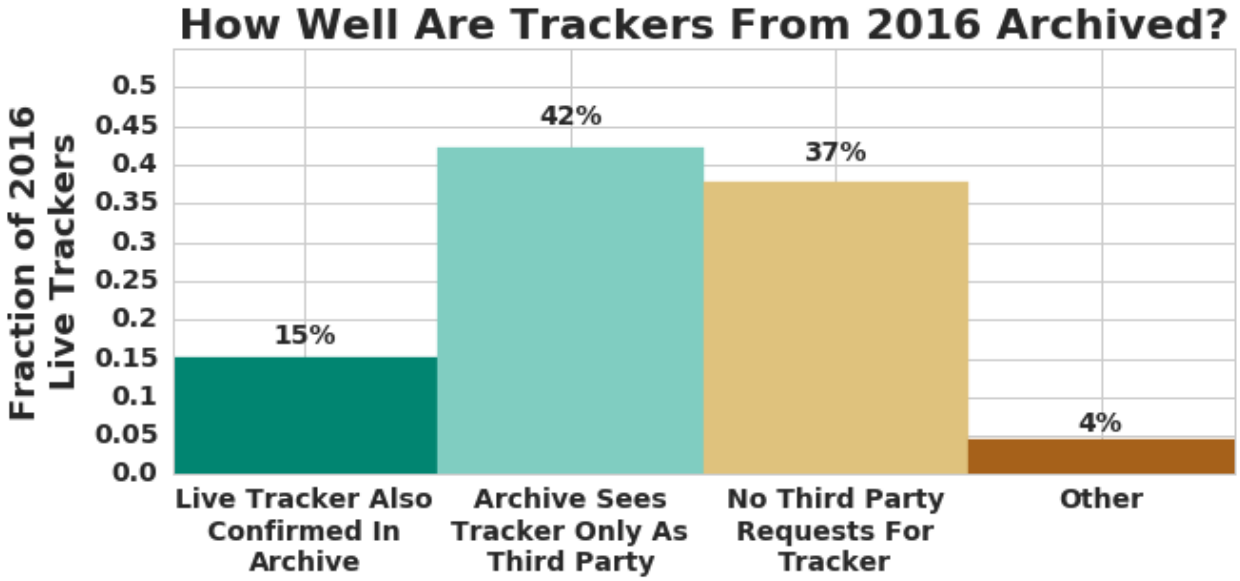



Figure 2.12: The fraction of domains categorized as Vanilla trackers in the live 2016 crawl which, in the archival 2016 crawl, (1) set and leaked cookies and thus were confirmed as trackers, (2) were only third-party requests (had at least one third-party request but no cookies), (3) did not appear at all, or (4) other (e.g., had cookies but not at the time of a third-party request, or cookies were not attached due to a cookie simulation bug).

**Cascading Failures.** Any of the above failures can lead to cascading failures, in that non-archived responses or blocked requests will result in the omission of any subsequent requests or cookie setting events that would have resulted from the success of the original request. The "wake" of a single failure cannot be measured within an archival dataset, because events following that failure are simply missing. To study the effect of these cascading failures, we must compare an archival run to a live run from the same time; we do so in the next subsection.

### *2.6.2 Lesson (Opportunity): Consider all third-party requests, in addition to confirmed trackers*

In the previous section, we evaluated the Wayback Machine's view of third-party requests *within* an archival measurement. For requests affected by the issues in Table 2.4, TrackingEx-

cavator observes the existence of these requests — i.e., counts them as third parties — but without the corresponding response may miss additional information (e.g., set cookies) that would allow it to confirm these domains as trackers according to the taxonomy presented earlier. However, this analysis cannot give us a sense of how many third-party requests are entirely absent from Wayback data due to cascading failures, nor a sense of any other data missing from the archive, such as missing cookie headers on otherwise archived responses. For that, we must compare directly with live results.

We focus our attention on unique trackers: we attempt to identify which live trackers are missing in the 2016 Wayback dataset, and why. For each tracker we observe in our 2016 live measurement, Figure 2.12 identifies whether we (1) also observe that tracker in "Wayback mode," (2) observe only a third-party request (but no confirmed cookie-based tracking behavior, i.e., we classify it only as a third-party domain), or (3) do not observe any requests to that tracker at all.

We conclude two things from this analysis. First, because the Wayback Machine may fail to provide sufficient data about responses or miss cookies even in archived responses, many trackers confirmed in the live dataset appear as simple third-party requests in the Wayback data (the second column in Figure 2.12). For example, `doubleclick.net`, one of the most popular trackers, appears as only a third party in Wayback data because of its `robots.txt` file. Thus, we learn that to study third-party web tracking in the past, due to missing data in the archive, *we must consider all third-party requests*, not only those confirmed as trackers according to the taxonomy. Though considering only third-party requests will overcount tracking in general (i.e., not all third parties on the web are trackers), we find that it broadens our view of tracking behaviors in the archive.

Second, we find that a non-trivial fraction of trackers are missing entirely from the archive (the third column in Figure 2.12). In the next subsection, we show that we can nevertheless draw conclusions about trends over time, despite the fact that the Wayback Machine under-represents the raw number of third parties contacted.

| | 2011 | 2013 | 2015 | 2016 |
|---|---|---|---|---|
| Wayback (All Third Parties) | 553 | 621 | 749 | 723 |
| Wayback (Vanilla+Personal) | 47 | 49 | 92 | 90 |
| Live (Vanilla+Personal) | 370 | 419 | 493 | 459 |
| Wayback-to-Live Ratio (Vanilla+Personal) | 0.13 | 0.12 | 0.19 | 0.20 |

Table 2.5: We compare the prevalence of the most common tracking types (Vanilla and Personal) over the four years for which we have data from the live web. Though the Wayback Machine provides only partial data on trackers, it nevertheless illuminates a general upward trend reflected in our ground truth data.

### *2.6.3 Lesson (Opportunity): The Wayback Machine's data allows us to study trends over time*

As revealed above, the Wayback Machine's view of the past may miss the presence of some third parties entirely. Thus, one unfortunately cannot rely on the archive to shed light on the exact raw numbers of trackers and other third parties over time. Instead, we ask: does the Wayback Machine's data reveal genuine historical *trends*?

To investigate trends, we compare all of our live datasets (2011, 2013, 2015, and 2016) to their Wayback counterparts. Table 2.5 compares the number of Vanilla and Personal trackers (the most prevalent types) detected in each dataset. For the purposes of this comparison, we sum the two types, since their distinction depends only on the user's browsing behaviors. We also include the number of all third parties in the Wayback datasets, based on the previous lesson. Though not all of these third parties represent trackers in live data, they help illuminate trends in third party prevalence over time.

We draw two conclusions from this comparison. First, we find that we *can* rely on the archive to illuminate general trends over time. Although confirmed trackers in "Wayback mode" (as expected from our earlier lessons) underrepresent the number of confirmed trackers found on the live web — and third parties in the archive overestimate confirmed trackers in the live data — we find that the trends we see over time are comparable in both sets of measurements. Critically, we see that *the upward trend in our archival view is not merely*

*the result of improvements in archive quality over time or other factors — we indeed observe this trend reflected in ground truth data.* We gain further confidence in these trends in Section 2.5, where we see a rise in tracking behaviors since 1996 that corresponds with our intuition. The absence of any large vertical steps in the figures in Section 2.5 further suggests that the trends we identify are artifacts of the web evolving as opposed to any significant changes in the Wayback Machine archival process.

Second, however, we find that — although long-term trends appear to be meaningfully represented by the Wayback Machine — one should not place too much confidence into *small* variations in trends. For example, the Wayback Machine's data in 2013 appears to be worse than in other years, under-representing the number of confirmed trackers more than average. Thus, in Section 2.5, we do not report on results that rely on small variations in trends unless we have other reasons to believe that these variations are meaningful.

### *2.6.4 Lesson (Opportunity): Popular trackers are represented in the Wayback Machine's data*

Because popular trackers, by definition, appear on many sites that users likely browse to, they have a strong effect on user privacy and are particularly important to examine. We find that although the Wayback Machine misses some trackers (for reasons discussed above), *it does capture a large fraction of the most popular trackers* — likely because the Wayback Machine is more likely to have correctly archived at least one of each popular tracker's many appearances.

Specifically, when we examine the 2016 archival and live datasets, we find that 100% of the top 20 trackers from the live dataset are represented as either confirmed trackers or other third parties in the Wayback data. In general, more popular trackers are better represented in Wayback data: 75% of the top 100 live trackers, compared to 53% of all live trackers. Tracker popularity drops quickly — the first live tracker missing in Wayback data is #22, which appears on only 22 of the top 500 websites; the 100th most popular tracker appears on only 4 sites. By contrast, the top tracker appears on 208 sites. In other words, those

trackers that have the greatest impact on user privacy do appear in the archive.

Based on this lesson, we focus part of Section 2.5's analysis in on popular trackers, and we *manually label* those that the Wayback Machine only sees as third parties but that we know are confirmed trackers in live data.

### *2.6.5 Lesson (Opportunity): The Wayback Machine provides additional data beyond requests*

Thus far, we have considered third-party requests and confirmed cookie-based trackers. However, the Wayback Machine provides, and TrackingExcavator collects, additional data related to web tracking behaviors, particularly the use of various JavaScript APIs that allow third parties to collect additional information about users and their machines (e.g., to re-identify users based on fingerprints). For JavaScript correctly archived by the Wayback Machine, TrackingExcavator observes accesses to the supported APIs (Appendix 2.7). For example, we observe uses of `navigator.userAgent` as early as 1997.

### *2.6.6 Summary*

In summary, we find that the Wayback Machine's view of the past is incomplete, and that its weaknesses particularly affect the third-party requests critical for evaluating web tracking over time. We identified and quantified those weaknesses in Section 2.6.1, and then introduced findings and strategies for mitigating these weaknesses in Sections 2.6.2-2.6.5, including considering third-party requests as well as confirmed trackers, manually labeling known popular trackers, and studying general trends over time instead of raw numbers. We leverage these strategies in our own measurements. By surfacing and evaluating these lessons, we also intend to help guide future researchers relying on data from the Wayback Machine.

We focus on the Wayback Machine since it is to our knowledge the most comprehensive web archive. Applying our approach to other, more specialized archives [144], if relevant for other research goals, would necessitate a new evaluation of the form we presented here.

## *2.7 Fingerprint-Related JavaScript APIs*

As described in Section 2.4, TrackingExcavator hooks a number of JavaScript APIs that may be used in fingerprint-based tracking and drawn from prior work [43, 134, 137, 12, 13]. The complete list:

- `navigator.appCodeName`
- `navigator.appName`
- `navigator.appVersion`
- `navigator.cookieEnabled`
- `navigator.doNotTrack`
- `navigator.language`
- `navigator.languages`
- `navigator.maxTouchPoints`
- `navigator.mediaDevices`
- `navigator.mimeTypes`
- `navigator.platform`
- `navigator.plugins`
- `navigator.product`
- `navigator.productSub`
- `navigator.userAgent`
- `navigator.vendor`
- `navigator.vendorSub`
- `screen.availHeight`

- screen.availLeft
- screen.availTop
- screen.availWidth
- screen.colorDepth
- screen.height
- screen.orientation
- screen.pixelDepth
- screen.width
- CanvasRenderingContext2D.getImageData
- CanvasRenderingContext2D.fillText
- CanvasRenderingContext2D.strokeText
- WebGLRenderingContext.getImageData
- WebGLRenderingContext.fillText
- WebGLRenderingContext.strokeText
- HTMLCanvasElement.toDataURL
- window.TouchEvent
- HTMLElement.offsetHeight
- HTMLElement.offsetWidth
- HTMLElement.getBoundingClientRect

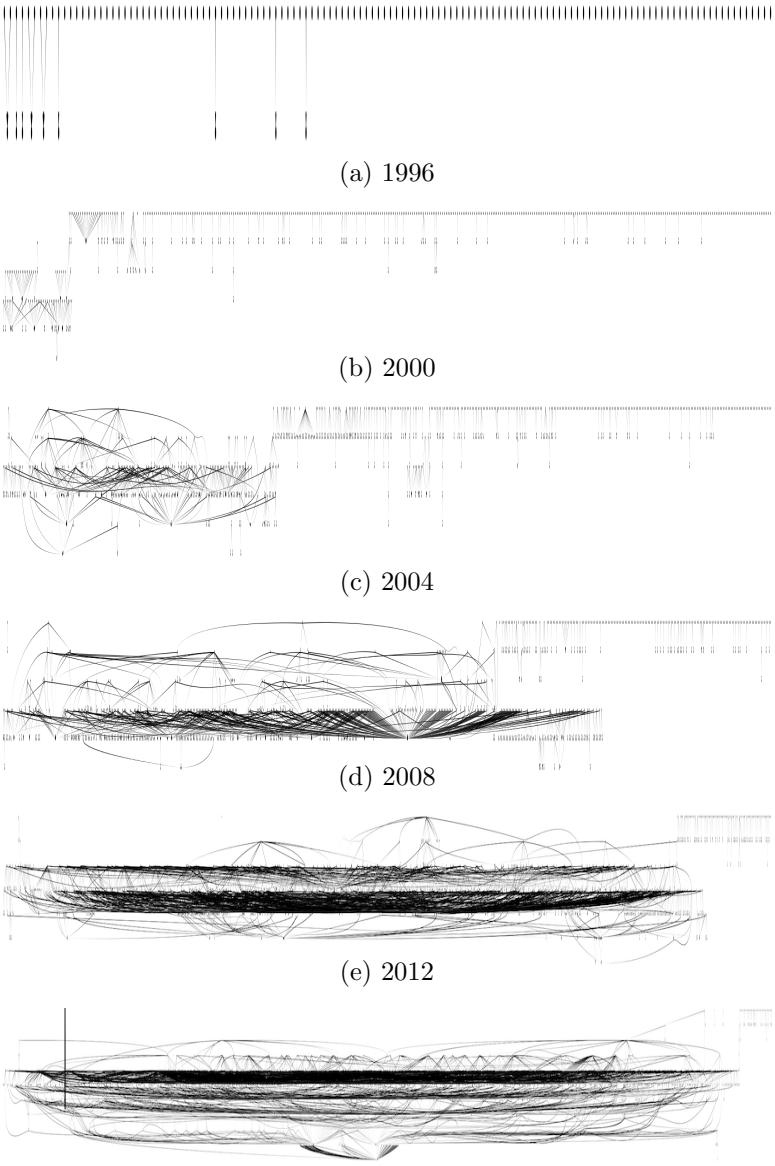
## *2.8 Ecosystem Complexity*

Figure 2.13 (on the next page) visually depicts the connections between entities in the tracking ecosystem that we observe in our datasets for 1996, 2000, 2004, 2008, 2012, and 2016: domains as nodes, and referral relationships as edges. Note that the visual organization of these graphs (with nodes in multiple tiers) is not meaningful and simply an artifact of the graph visualization software. Over time, the complexity and interconnectedness of relationships between third-party domains on the top 450 websites has increased dramatically.

## *2.9 Conclusion*

Though third-party web tracking and its associated privacy concerns have received attention in recent years, the practice long predates the first academic measurements studies of tracking (begun in 2005). Indeed, in our measurements we find tracking behaviors as early as 1996. We introduce TrackingExcavator, a measurement infrastructure for third-party web tracking behaviors that leverages `archive.org`'s Wayback Machine to conduct historical studies. We rigorously evaluate the Wayback Machine's view of past third-party requests and develop strategies for overcoming its limitations.

We then use TrackingExcavator to conduct the most extensive longitudinal study of the third-party web tracking ecosystem to date, retrospectively from 1996 to present (2016). We find that the web tracking ecosystem has expanded in scope and complexity over time: today's users browsing the web's popular sites encounter more trackers, with more complex behaviors, with wider coverage, and with more connections to other trackers, than at any point in the past 20 years. We argue that understanding the trends in the web tracking ecosystem over time — provided for the first time at this scale by our work — is important to future discussions surrounding web tracking, both technical and political.



(a) 1996

(b) 2000

(c) 2004

(d) 2008

(e) 2012

(f) 2016

Figure 2.13: Referrer graphs for the top 450 sites in 1996, 2000, 2004, 2008, 2012 and 2016 as seen in the Wayback Machine's archive. An edge from a domain `referrer.com` to another domain `referred.com` is included if any URL from `referrer.com` is seen to be the referrer for any request to `referred.com`. Note the increasing complexity of the graph over time.

# Chapter 3

# REWRITING HISTORY: CHANGING THE ARCHIVED WEB FROM THE PRESENT

In this chapter, I pivot from using sister technologies as a source of measurement data (as in Chapter 2, where the Wayback Machine was used to study web tracking and other properties of the web) to considering those sister technologies directly. I consider the socially important uses of web archives (such as in academic research, journalism, and legal proceedings) and study how motivated attackers might maliciously manipulate the view that clients have of archival content. After discovering and developing vulnerabilities and attacks, this work uses TrackingExcavator (Chapter 2) to quantify the prevalence of these vulnerabilities, demonstrating the flexibility and power of this new measurement tool.

The work presented in this chapter is currently under submission, and if available, should be cited in its conference-paper form.

## *3.1 Introduction*

The Wayback Machine is a publicly browsable web archive which has cataloged and preserved a collection of over 286 billion web pages over the period from 1996 to 2017 [76]. Like other web archives, the Wayback Machine allows clients using ordinary web browsers to access snapshots of past websites through a web interface[1], enabling ordinary citizens as well as technical experts to see how the web has changed and what it once contained. These archival snapshots of websites are rendered in HTML, Javascript, and CSS just like the modern web, preserving not only their content but also their client-side dynamic behaviors, making them a rich cultural and technical preserve.

---

[1] `https://web.archive.org/`

The Wayback Machine is frequently used in a variety of contexts critical to our free society, including scholarly articles, journalism, and legal proceedings. Scientists may cite archived snapshots in their scientific papers to increase the durability of their references [42, 143], while journalists have used archives to understand how websites such as official government pages have changed [139], and lawyers often use archival snapshots as evidence in legal cases, including civil and criminal cases, administrative proceedings, and patent litigation (e.g., [3, 4, 6, 142]). While other researchers have studied inaccuracies in the Wayback Machine which arise accidentally, we observe that these socially and financially important uses suggest incentives to *intentionally* manipulate archives after the fact. For example, governments might want to suppress or change historical information, companies might want to manipulate evidence of prior art in a patent case, organizations might want to hide evidence of past wrongdoing, and news sources might want to manipulate source material for their reporting.

To our knowledge, this chapter describes the first work investigating the technical vulnerabilities and attacks that might be used to perform such intentional manipulation. That is: how might attackers attempt to rewrite history? How might they intentionally cause clients who view the archive to see archived websites with content, appearance, and behavior that are different from the actual website at archival time? We analyze the way that the Wayback Machine functions, finding that in fact, there are several types of vulnerabilities which would allow an attacker today to take full control of clients' views of snapshots. For example, snapshots sometimes cause clients to accidentally mix content from the live web into an archived page, allowing servers on the live web to inject content or code into clients' views of the archive. Our attacks are global — they affect the appearance and behavior of snapshots for all visitors, and they do not involve the direct compromise of archival or publisher servers or databases.

We demonstrate the viability of our attacks with proofs-of-concept. For example, we demonstrate the ability to inject arbitrary Javascript code into client views of archival snapshots, allowing us to modify text, images, styling, and behavior, subtly or completely rewriting the web of the past. Figure 3.1 shows such an attack, in which we took complete control

of a snapshot of `reuters.com` from 2011.[2]

We then quantify the prevalence of the types of vulnerabilities we discovered, seeking them in the wild through a measurement study of archived websites. We find that vulnerabilities to our attacks are very common: over snapshots of the Top 500 most popular websites of the past 20 years, 74% contain some vulnerability which exposes the snapshot to complete control by an attacker (65% for URLs sampled from the Top Million). Additionally, we perform these same measurements over a set of website snapshots which have been cited in legal contexts such as court decisions, administrative decisions, and documents filed as trial court and appellate briefs, finding that 37 domains referenced in the 991 legal documents we examined are vulnerable to an attack which would provide complete control to some attacker over the way clients view the snapshot. We note that we are unaware of any attackers who have used these vulnerabilities for malicious purposes in practice — rather, our measurements show that a large fraction of sites are or were vulnerable to such attacks, suggesting that the consumer of web archives should exercise caution.

While an instance of our attacks may be evident upon detailed technical inspection of the way a client renders a snapshot, they are likely to be completely invisible to less technical users of the archive. Even when investigated by technical experts, attackers may have plausible deniability, since modern content can and does become intermingled with archival content in many benign cases [89, 27]. We explore a variety of defenses that could help clients see correct views of snapshots, and we design and build ArchiveWatcher, an end-user defense which demonstrates a subset of our defensive techniques, demonstrating techniques which may aid technical experts, such as expert witnesses and fact checkers in legal and journalism contexts, in determining when a view of a website can be reliably cited.

This chapter makes the following contributions:

- We analyze the Wayback Machine in order to identify vulnerabilities which enable adversaries to manipulate clients' view of archival snapshots (Section 3.4).

[2]For ethical reasons, we disabled our attacks after showing that they worked.

- We develop attacks which exploit these vulnerabilities, exploring how an adversary can change the appearance and behavior of snapshots seen by all visitors to the archive, even years after the snapshot was captured. We execute proofs-of-concept of our attacks against real snapshots in the Wayback Machine (Section 3.5).
- We measure the prevalence in the wild of vulnerabilities which enable our attacks, finding that they are quite common, including a number of vulnerabilities which affect snapshots cited in legal cases and decisions (Section 3.6).
- We explore the space of possible defenses which might be deployed by archives, website publishers, and end-users, and we build an end-user defense, ArchiveWatcher, that detects and blocks vulnerabilities to our attacks (Section 3.7).

Before the publication of the conference paper describing this work, we will disclose these vulnerabilities to the Wayback Machine, and we will make our defense, ArchiveWatcher, publicly available.

## *3.2 Background and Related Work*

### *3.2.1 How Web Archives Work*

**Overview: Archival Protocol and Systems.** We focus our analysis of web archives on the Internet Archive's Wayback Machine, since it is the largest publicly available web archive, with a goal of archiving as much of the public available web as possible.

The Wayback Machine consists of two major components relevant to this paper. The first is the **archive crawler**, which visits, retrieves, loads, and archives pages on the web into the archive's database. The second is the **archive front-end**, which is the system of web servers, accessible via `https://web.archive.org`, which allow anyone to use their browser to view the web of the past.

In this paper, we refer to the archival preservation of a top level page as an **archival snapshot**, or simply **snapshot**, and the archival copies of a page's subresources (e.g., images, scripts, CSS, etc.) as **archival captures**. Each snapshot or capture was saved at a moment

in time, called its **timestamp**, which appears in its URL. For example, `https://web.archive.org/web/20001110101700/http://www.ccs2000.org:80/` refers to a capture of the homepage page for the 7th CCS which was saved by the archival crawler at 10:17:00 UTC on 11 November 2000. When a web browser visits this snapshot, it does the same thing as when it accesses a normal site on the live web: it recursively downloads, parses, executes, and renders the HTML, Javascript, and CSS of the page. The only difference is that the archive plays the role of the first- and third-party web servers which originally published the the site, serving the resources that make up the snapshot.

The archive crawler performs regular crawls of a large set of pages, providing significant coverage of the web. Internet Archive's Frequently Asked Questions page does not offer details about how they find sites to crawl, but states that "crawls tend to find sites that are well linked from other sites", and that they collect pages that are "publicly available" [77]. Additionally, any person can use a form on the Wayback Machine's website **"Save Page Now"**, which "Capture[s] a web page as it appears now for use as a trusted citation in the future." This feature causes the archival crawler to immediately capture the given page or resource, including its subresources [78]. We discuss additional technical details about the Wayback Machine inline as appropriate.

### *3.2.2 How are Web Archives Used?*

Web archives are used in variety of important social contexts, including legal proceedings, news articles, academic publications. We take particular interest in their use in legal proceedings for two reasons: because the integrity of the legal process is important to our free society, and because legal proceedings may motivate involved parties to launch attacks that modify evidence in their favor, such as by using the attacks described in this paper. Lawyers use web archives in a wide variety of legal contexts, such as civil lawsuits (e.g., [6]), criminal cases (e.g., [4]), administrative proceedings (e.g., [5]), federal claims court (e.g., [3]), and patent litigation (e.g., [142]), anc they may use archival evidence for various purposes, such

as to demonstrate "prior art" in patent litigation [3] or to recover evidence of wrongdoing that has since been deleted from the live web.

Because of these socially important uses, users of archives should take appropriate steps to ensure that archival data they use is trustworthy and not manipulated. We emphasize that we are unaware of any attacks like the ones in this paper being used in practice. However, this work demonstrates that not only are attacks are possible (Sections 3.4 & 3.5), but also that the vulnerabilities which enable them are very common in the wild (Section 3.6).

### *3.2.3 Legal Guidance on Web Archives*

Legal scholars have written on the evidence standards that do and should govern the admissibility of archival material. Eltgroth encouraged the use of existing evidence standards to allow "reliable evidence from the Wayback Machine [to be] admitted as any other Internet-derived proof" [48], while Gazaryan argued in 2013 argued for the need to lower the difficulty of using archival material as evidence [57]. Others have advised lawyers on best practices such as employing experts to evaluate the technical limitations of the archive [142]. These articles discuss only non-adversarial factors, while we focus on the technical aspects of adversarial manipulation rather than the legal aspects of incidental inaccuracies.

In 2007, Fagan raised the possibility of "E-Evidence Tampering", noting that archival infrastructure may be compromised, or that an archived website might be cached or archived in a compromised state [53]. Our work is different in that we consider less privileged attackers, who do not compromise the archive.

### *3.2.4 Technical Work on or with Web Archives*

Computer scientists have used the Wayback Machine in research: Nikiforakis et al. measured longitudinal trends in Javascript inclusion from 2001 to 2010 [135]; Soska and Christin used archival data to develop and evaluate a method for determining which websites would become

[3] Patents must be original to be valid, and prior art is information published prior to a patent which might be relevant to the patent's claims of originality[7].

malicious over time [158]; Lerner et al. studied third-party web tracking using archival data [14]; and Hackett et al. studied the evolution of website accessibility from 1997 to 2002 [65].

Others have studied the (non-malicious) incompleteness or inconsistency of web archives (e.g., [130, 88, 18, 28]). We find in our work that the technical limitations of archives that lead to accidental incompleteness can be leveraged intentionally by adversaries.

## *3.3 Threat Model*

In our threat model, we consider attacks in which clients (both people and automated systems) browsing archival material are maliciously caused to see content that does not accurately reflect the the web of the past. Critically, we show that this is possible *without* requiring attacks to be launched by the archive itself, and *without* compromising website publisher or archival servers. Instead, the vulnerabilities which enable our attacks involve entirely ordinary interaction with archives, such as hosting content on domains and servers the attacker rightfully owns and requesting that the archive capture specific URLs.

We note that the vulnerabilities we consider can also cause non-malicious inaccuracies in the archive. These non-malicious inaccuracies have been discussed in other work (e.g., [17, 16, 151]), and could an our defenses Section 3.7 might incidentally mitigate them. However, we focus on the ways in which our vulnerabilities can be used intentionally by malicious actors.

### *3.3.1 Definitions*

We refer to a single capture of a web page as a **snapshot** or **archival snapshot**. For example, `http://web.archive.org/web/20000101000000/http://example.com` is a snapshot of `http://example.com` which aims to represent its appearance as of 1 January, 2000. We will use the terms **time-of-archive**, **timestamp**, or **archival timestamp** to refer to the time at which a particular snapshot was taken. Prior to time-of-archive, we may refer to **time-of-publication**, when the first-party website chose what content to include in its web-

site and published it on the web. We may use these terms to refer to the domains involved in an attack and their owners at different times. For example, we may refer to the time-of-archive first-party, by which we mean "the entity which owned `example.com` at the time that the snapshot in question was archived," noting that ownership may change over time. Figure 3.2 depicts the relationship of different times in the lifecycle of a snapshot.

We will refer to as **clients** the end-users and devices that use the archival front-end to view snapshots, and who may rely upon those snapshots for information about the past. For example, a client may wish to refer to the content of `http://example.com` in 2000 in the course of a legal argument. To do so, they would use an ordinary browser (the **client browser**) to access the snapshot "`http://web.archive.org/web/20000101000000/http://example.com`". We will refer to the time at which a client accesses a snapshot as the **time-of-access**. For example, if a client examines the past contents of `example.com` on 19 May 2017, then 19 May is the time-of-access in this scenario. If an attack has been made against that snapshot, then the client may see a modified version of the snapshot at the time-of-access, rather than something which accurately reflects the site's appearance at time-of-archive.

We refer to the time at which an attacker takes an action to deploy an attack as the **time-of-attack**. Since our attacks sometimes require multiple actions by the attacker at different times, there may be multiple times-of-attack for a scenario. The time-of-attack may be either before or after time-of-archive, depending on the attack, and time-of-attack may precede or coincide with payload delivery to the client at time-of-access.

### *3.3.2 Attacker's Goals*

Our attacks aim to change what clients see when they view archived snapshots — that is, to cause the client browser to display snapshots incorrectly, rendering content and exhibiting behavior (i.e., running code) which do not reflect the original website nor (in the case of benign archival errors) the website as it had originally been preserved in the archive.

We observe that attackers may have incentives to modify both their own and others'

content in the archive. For example, if Alice accuses Bob of publishing slander on his website, then Bob may wish to retroactively remove the slander from the archive of his website. Alternatively, Alice (or an uninvolved party, such as Mallory) may frame Bob by retroactively adding slander to snapshots of his site. Attackers may be motivated by a wide variety of personal, political, legal, and financial motivations.

We emphasize that although our threat model encompasses attacks that add material to the archive's databases, the adversary must only do so legitimately, *not* by compromising those databases. That is, some attacks involve archiving new websites that we create as part of an attack.

By default, successful attacks are visible to *any* client who views that archived resource. However, attackers could also customise their attacks for different clients. For example, attackers might identify clients via techniques like browser fingerprinting [54, 138, 44, 128], or by using tracking cookies [147]. Though we note such customization is possible, we do not explore it further in this chapter.

### *3.3.3 Possible Attackers*

Under our threat model, the attacker owns — at time-of-attack — the domain from which the attack is mounted. For a given victim snapshot, the attacker may either be the owner of the **first-party** domain (e.g., `example.com`) or the owner of a **third-party** domain on that page (e.g., `ads.com`, serving an ad embedded inside `example.com`).

In a third-party attack, an attacker who controls `ads.com` (either at time-of-archive or in the future) may wish to modify the snapshot of `example.com`. To motivate a first-party attack — `example.com` modifying itself — we note that the ownership of domains may change over time. Thus, for example, a different entity may own `example.com` now than in the past, and that new owner may now wish to modify past archives of `example.com`. The present first-party owner might also be the same as the past owner, but seeking to alter its own past archives.

Thus, depending on the attack, an attacker must be able to serve content from one of

the first- or third-party domains that make up the target snapshot, either at time-of-archive and/or at time-of-access. To meet this criterion, the attacker may either already own relevant domains, or they might purchase domains specifically to perform these attacks. They might also be able to hijack domains illicitly, e.g., through DNS poisoning. The means by which the attacker gains the ability to publish content from the domain of the vulnerable resource is orthogonal to the discussions of this chapter.

## *3.4 Analyzing the Wayback Machine for Vulnerabilities*

We analyzed the Wayback Machine, surfacing three types of vulnerabilities which emerge from its design. Those types of vulnerabilities are **Archive-Escapes**, **Same-Origin Escapes**, and **Never-Archived Resources**, detailed below.

### *3.4.1 Archive-Escapes*

To deliver snapshot content, the Wayback Machine plays the role of all web servers which were originally involved in serving the archived site. That is, it serves archived versions of all first- and third-party content the client requests while rendering its view of the snapshot. To cause the client to correctly request all these resources from the archive, rather than the live web, the archive performs **URL rewriting**, modifying URLs in archived HTML, Javascript, and CSS to make them refer to archived versions of the same URL. For example, the archive may find the URL `http://example.com/script.js` in some HTML at time-of-archive, and rewrite the HTML so that the URL instead reads `http://web.archive.org/web/<timestamp>/example.com/script.js`, where the timestamp of the archived script matches the timestamp of the archived HTML.

URL rewriting is not perfect, primarily because it does not account for client-side dynamically generated URLs. We find that when Javascript computes subresource URLs using computation as simple as string concatenation, then URL rewriting fails and *clients end up making requests to the live web* to load those subresources. For example, if URL rewriting fails, the client might accidentally load a live copy of `example.com/script.js` instead of

its archived version. These live web subresources are incorporated into the client's rendered view of the snapshot, mixing live and archived content and behavior.

We refer to the request and use of live-web resources as part of a snapshot view as an **Archive-Escape**, the first of our classes of vulnerabilities. We refer to the domain contacted for live resources as the **archive-escape destination**, such that in the example above, `example.com` is an archive-escape destination. Whenever there is an archive-escape, the destination of that escape becomes a potential attacker, since that domain can now serve a malicious payload on the live web at the escaping URL. For example, the live copy of `example.com/script.js` can be replaced with a malicious payload. Note that the archive-escape destination may be the same domain as that of the victim snapshot.

### *3.4.2 Same-Origin Escapes*

We discovered a second class of vulnerability, related to the fact that archives take on the role of serving both content from *all* of the domains which were involved in a snapshot at time-of-publication.

As background, browsers prevent third-parties inside `<iframe>`s from accessing or modifying data from the main page. This policy of preventing cross-origin access is called the Same-Origin Policy. So, for example, if `http://example.com` embeds `http://ads.com` in a frame, code from `ads.com` (running inside the frame) will be blocked by the browser from reading or influencing any parts of the page outside of its frame. This allow sites to safely embed content from third-parties within the context of their own pages. The `http://ads.com` attacker might embed malicious code which attempts to modify the page, but it will be blocked from doing so by the Same-Origin Policy.

The Same-Origin Policy, however, is ineffective in the archival context. Since all archived resources are loaded from the archive, this means that *all* resources making up a snapshot, including both first- and third-party resources, are loaded by the client from *a single domain*, `archive.org`. When this occurs, a vulnerability arises: code from the embedded frame now executes without the isolation provided on the live web by the Same-Origin Policy, allowing it

to reach outside of its frame to modify any aspect of the main page. This allows an attacker to embed an attack payload inside of an `<iframe>`, where it will become active when preserved by the archive and served to clients, modifying the client's view of the containing snapshot.

### *3.4.3 Never-Archived Resources and Nearest-Neighbor Timestamp Matching*

Our third class of vulnerability arises from the interaction of two properties of the Wayback Machine: its incompleteness, and its nearest-neighbor timestamp matching.

First, we discuss incompleteness. Many pages in the Wayback Machine include resources which the archive has never successfully captured. There are a variety of reasons why this might occur, including archival crawler errors or a partial unavailability of the publisher's web server at time-of-archive. For example, snapshot's HTML might include an image, but that image has never been saved in the archive's database. When the client asks for a never-archived resource, the archive front-end responds with an HTTP `X-Archive-Wayback-Runtime-Error` header with value `ResourceNotInArchiveException`, and error code 404. Our measurements (Section 3.6 show that never-archived resources arise quite commonly.

Second, we discuss the archive front-end's **nearest-neighbor timestamp matching** policy. Image that a client requests an archived resource $R$ at a timestamp $T$, and that the archive's database contains captures of $R$, but only with timestamps $\neq T$. When this happens, the archive will find the capture of $R$ with timestamp as close as possible to $T$, and redirect the client to that version. For example, imagine a client that requests to visit a March 2005 snapshot of `example.com`. If `example.com` was never captured in March of 2005, but was captured in April, then the archive would redirect the browser (302 FOUND) to the April timestamp.

In non-malicious situations, this "nearest-neighbor" behavior allows clients to view a more complete picture of the past in the case that a snapshot's subresources were not captured at the exact moment the snapshot was. However, there is no apparent limit to the time delta permitted by nearest-neighbor timestamp matching. Thus it is possible, for example,

to request a resource from 1996 and be redirected to a capture of that resource from 2016, if no other closer timestamp exists. We refer to instances where client browsers are redirected to timestamps very far in time from the original page as **anachronisms**.

An attacker who owns the domain of a never-archive resource can abuse these observations by inserting a malicious payload as the anachronistic capture of that missing resource, which will be served to clients due to nearest-neighbor matching.

## *3.5 Rewriting History: Our Attacks*

Having discussed our vulnerabilities, we delve into the design of attacks which exploit these vulnerabilities to rewrite history. For reference in discussing these attacks, recall that Figure 3.2 depicts the lifecycle of a snapshot and possible attacks against it.

### *3.5.1 Attack #1: Archive-Escape Abuse*

**Preliminaries and Attacker.** The precondition for Archive-Escape Abuse is the presence of an archive-escape vulnerability in the victim snapshot. The potential attacker is the owner of the destination of the archive-escape, to whom the client makes a request for the vulnerable resource. Because the attacker delivers the payload from their own servers (rather than via the archive) at time-of-access, we refer to this as an **active** attack.

**Attack Concept.** To mount this attack, the attacker (the destination of an archive-escape), publishes malicious content at the escaping URL. If the archive-escape is to a static resource like an image, then the attacker will only be able to affect that resource; if the archive-escape is a request for a script or stylesheet, then the attacker can choose arbitrary malicious code to execute.

**Sequence of Events for Attack #1.**

1. The victim page is published. (Optional: If the attacker is the first-party domain wishing to enable future modifications of itself, the attacker can *intentionally* include requests which will result in archive-escapes.)

2. The page is archived as the victim snapshot.
3. The victim snapshot, when loaded, causes the client browser to make an archive-escape request.
4. The attacker (who owns the domain on which the escaping script is hosted) serves malicious code in response to the archive-escape request. The malicious code runs in the client browser and modifies the appearance of the snapshot so that the client sees an inaccurate view of the page.

**Proof of Concept Attack Implementation.** We developed a proof-of-concept implementation of Attack #1, demonstrating the ability to attack snapshots of websites over which we have no control and which were archived years ago. We used our measurements (Section 3.6) to locate archive-escape vulnerabilities where the attacker domain was unowned, using `whois`. Finding that `http://web.archive.org/web/20110901233330/reuters.com` generates an archive-escape to `http://cdn.projecthaile.com/js/trb-1.js`, and that as of 19 March 2017, `projecthaile.com` had no owner. We purchased `projecthaile.com` and hosted our own version of `/js/trb-1.js` which modifies specific elements of the `reuters.com` snapshot. This attack resulted in the screenshot shown in Figure 3.1, in which we replaced a news article image and headline with our own.

As with all of our attacks against snapshots we do not own, we disabled the attack after confirming that it worked, so as not to disrupt the public's view of the snapshot. Additionally, we have purchased the remaining unowned domains (without hosting anything from them) for this attack to prevent any other attackers from buying and using them.

**Advantages and Disadvantages of Attack #1.** Attack #1 is an **active** attack, where the attacker's server delivers payloads directly to clients, allowing an attacker to modify their attack over time, customize it per client, or disable the attack entirely. However, it also means the attack is not permanent. Additionally, defenses which block archive-escapes are among the easiest for clients to deploy.

### *3.5.2 Attack #2: Same-Origin Escape Abuse*

**Preliminaries and Attacker.** Potential Same-Origin Escape attackers include all third-parties embedded in `<iframes>` at time-of-archive. However, this attack requires foresight — the attacker needs to have included their payload inside their `<iframe>` at time-of-archive, so that it can be preserved and served from the archive's database. Note that this makes Attack #2 a passive attack, since the payload is stored and delivered to the client by the archive, rather than directly from the attacker's server at time-of-access.

**Attack Concept.** As described above, this attack abuses the lower level of isolation which the client browser applies to frames when they are delivered from a single origin (the archive's origin) rather than multiple origins, as they are served on the live web. The first-party publisher includes the attacker in their page under the assumption that malicious code the attacker writes to deface the first-party's page will be unable to do so because of the Same-Origin Policy, and this assumption is violated in the archival context.

**Sequence of Events for Attack #2.**

1. A victim site includes a third-party in an `<iframe>`, where they are now a potential Same-Origin Escape attacker.
2. The third-party attacker publishes malicious code in its `<iframe>`.
3. In the live web, the malicious code executes, but its effects are blocked by the browser, according to the Same-Origin Policy.
4. The first-party page is archived as a snapshot, including the attacker's `<iframe>`.
5. When the snapshot is loaded, both the page and the `<iframe>` are served from `web.archive.org`. Since they are now served from the same domain, the Same-Origin Policy no longer applies, and the malicious code in the `<iframe>` can make arbitrary modifications to client's view of the page.

**Proof of Concept Attack Implementation.** For Attack #2, we developed a prototype demonstration against a toy website which we created and archived for demonstration purposes. The reason is that this attack requires the attacker to be a third-party with foresight,

and we do not have a third-party position on any websites we do not control which we could use to demonstrate the attack.

Thus, to demonstrate this attack, we published, on the live web, the victim page of our first-party domain, including an `<iframe>` of our third-party domain. Inside the `<iframe>`, we then deployed attack code which attempts to modify elements of the first-party page. On the live web, this attack code fails, due to the Same-Origin Policy. We then requested that the Wayback Machine "Save Page Now" for our first-party victim page, causing it to archive that page and, as part of archiving that page, also archive the attacker's `<iframe>` with its attack code. When viewing the snapshot of the victim page in the archive, both first- and third-party content are served from the same domain, causing the Same-Origin Policy to no longer apply, and allowing the third-party code to modify clients' views of the victim snapshot.

**Advantages and Disadvantages of Attack #2.** This attack has several strengths. First, the prerequisites for performing the attack are minimal, since all that is required is to be a third-party who can execute Javascript. Third-party frames are commonly embedded and trusted by websites, and it may even be possible to purchase advertising space in order to gain the position needed to execute the attack. Additionally, there are some third-parties who are present on a large fraction of websites (see Section 3.6), meaning that for certain attackers, this attack represents a huge capability to modify snapshots of a large number of websites.

However, this attack is significantly limited because the attacker must have **foresight**: Their attack code, and thus the changes they wish to cause in the client's view, must be chosen before time-of-archive, since the attack code must itself be stored in the archive.

### *3.5.3 Attack #3: "Same-Origin Escape" + "Archive-Escape"*

**Preliminaries and Attacker.** Noting the limitation of Attack #2 requiring foresight, we consider a stronger way to use Same-Origin Escapes: Attack #3. This attack uses a Same-

Origin Escape to create an intentional archive-escape, allowing the attacker to launch a later attack without foresight. Attack #3 is applicable any time Attack #2 is applicable, since it begins with a third-party in an `<iframe>` executing Attack #2 in order to create a later opportunity for Attack #1.

**Attack Concept.** This attack combines Attacks #1 and #2. Here, the attacker uses a Same-Origin Escape (malicious code in an `<iframe>`) to intentionally cause archive-escapes, with a destination the attacker controls, in the snapshot of the victim page. Once this has been done, the attacker is now capable of performing archive-escape abuse, immediately or at a later time.

**Sequence of Events for Attack #3.**

1. The attacker must be a third-party who is embedded as an `<iframe>` on the target page as of time-of-publication.
2. The attacker chooses a destination payload URL which they control, and embeds an archive-escape to that URL as the `src` attribute of a `<script>` tag in their `<iframe>`.
3. The page, along with the `<iframe>`, is archived.
4. Some time in the future, the attacker chooses and publishes a payload at the archive-escape URL.
5. When a client browser loads the snapshot, the archived `<iframe>` is retrieved from the archive, including the script which causes an archive-escape. The browser retrieves the payload and executes it in the context of the `<iframe>`. Since the `<iframe>` is archived, it is not isolated by the Same-Origin Policy (see Section 3.5.2) allowing the modern attack script to cause arbitrary modifications to the client's view of the snapshot.

**Proof of Concept Attack Implementation.** Since Attack #3 leverage Attack #2 (Same-Origin Escape), we created a similar victim/attacker pair of testbed websites to demonstrate this attack. We again deployed attack code inside a third-party `<iframe>`, but in this case our attack code used string concatenation to create an archive-escape to the third-party domain rather than directly modifying the snapshot content directly. We then

hosted the snapshot-modifying code on the live web at the third-party domain.

**Advantages and Disadvantages of Attack #3.** This attack allows archive-escape attacks against a page which does not naturally generate any archive-escapes to the attacker's domain, making it subject to the disadvantages of archive-escape attacks discussed above.

Since the archive-escape payload can be chosen after time-of-archive, this attack reduces a Same-Origin Escape attacker's need for foresight: they must only choose to enable a future attack by embedding a small amount of archive-escape generating code in the `<iframe>`, without the need to know how exactly they will change the snapshot in the future. An attacker such as a content delivery network or advertiser which appears on many pages could even choose to seed many pages with archive-escapes in order to preserve their ability to attack snapshots of many pages later on.

### *3.5.4 Attack #4: Anachronism-Injection*

**Preliminaries and Attacker.** The precondition for Anachronism-Injection is a page which contains at least one resource which has *never* been captured by the archive. The potential attacker is the owner of the domain of that never-archived resource, who is in a position to publish a malicious version of that resource and cause that payload to be preserved in the archive as the resource's first (and at that point only) capture.

**Attack Concept.** The attacker publishes payload code to the missing-resource's URL on the live web, then uses the archive's "Save Page Now" feature to archive the payload. For example, a snapshot from 2000 might include a script capture, also from 2000. If that script has never been archived, then today, in 2017, the owner of the script's domain can publish a malicious payload at the script's URL and use the archive's "Save Page Now" feature to create a capture of the script with a 2017 timestamp. Once the missing resource has been archived, it will be the only capture of that resource in the archive (since a precondition of the attack was that the resource had *never* before been archived). As the only capture of the resource, its timestamp necessarily is (and always will be) the nearest neighbor to

the timestamp requested in the victim snapshot, despite being 17 years distant. Thus the payload will be loaded in the context of the victim snapshot, as client requests are nearest-neighbor redirected to the malicious payload's timestamp. Even if more captures of the malicious resource are made afterwards, the payload will always have a timestamp that is strictly earlier, and thus which is closer to the victim snapshot's timestamp, than those subsequent captures, making the attack permanently effective.

**Sequence of Events for Attack #4.**

1. A victim snapshot refers to a vulnerable resource which has never been archived.
2. The attacker, who owns the vulnerable resource's domain, publishes an attack payload on the live web.
3. The attacker uses the archive's "Save Page Now" feature to cause the payload to be preserved as the first and only extant capture of the vulnerable resource.
4. When a client browses the victim snapshot, their browser makes a request for the vulnerable resource at the timestamp of the snapshot. In response, the archival front-end redirects the client browser to the malicious, anachronistic capture of the resource, since it has the timestamp closest to the requested version.

**Proof-of-Concept Attack Implementation.** As with Attack #1, we *could* demonstrate the Anachronism-Injection attack on snapshots of previously-archived websites over which we have no control. However, because this attack permanently alters the victim snapshot (even if our injected anachronism is not expressly malicious), we chose not to implement this attack on real victim snapshots. Instead, we test it on our own testbed websites, similarly to Attacks #2 and #3.

We note that executing this attack took careful planning, since on several occasions we deployed attack code that was slightly incorrect, forcing us to start over with entirely new victim and attacker pages, since once the attack code is archived, the attacker is unable to replace it with different attack code, since all subsequently archived code will have a timestamp farther from the victim snapshot's timestamp. However, using this attack we

were able to take control of our testbed victim snapshot.

**Advantages and Disadvantages of Attack #4.** This attack is a passive attack, with the advantage that once the attack is in place, it becomes permanent. However, the flip side to this advantage is that the attacker cannot easily disable the attack, since the content which enables the attack has been permanently preserved in the archive's database.

Indeed, this attack's main weakness is that it is a one-time opportunity. Once the attacker has created a payload and caused it to be archived, they no longer have any way to change the behavior of that attack, since it is permanently the closest neighbor to the vulnerable resource. However, an attacker could choose to make two distinct modifications to the attack to gain the ability to continue to modify the payload over time:

1. **Archive-escape extension**. In this version of the attack, the malicious code creates an intentional archive-escape, allowing persistent control from the present by the attacker. This version fails against archive-escape-blocking defenses.
2. **Anachronism chaining**. In this version, in addition to performing malicious modifications of the snapshot, the payload also causes the client to make a request for the archival version of another, different URL which has never been archived. In other words, while deploying the payload, the attacker intentionally creates the preconditions for another Anachronism Injection attack, which they can exploit in the future. For example, the archived payload script `attack0.js` might make a request for the never-archived script `attack1.js`. This request will fail until the attacker changes the content of the snapshot again, at which point they host and archive `attack1.js`. This chaining can continue indefinitely (`attack2.js`, `attack3.js`, etc.).

### *3.5.5 Reflecting on Attacks*

We now step back and reflect on our attacks, which are summarized in Table 3.1. We highlight several axes along which we can distinguish our attacks:

**Passive vs. Active Attacks.** Attacks differ by whether the payload is loaded from the

| # | Name | Requires Foresight? | Passive or Active? |
|---|---|---|---|
| 1 | Archive-Escape Abuse | No | Active |
| 2 | Same-Origin Escape | Yes | Passive |
| 3 | Same-Origin Escape ->Archive Escape | Yes | Active |
| 4 | Anachronism Injection | No | Passive |

Table 3.1: A summary of the attacks we develop. Attacks requiring foresight necessitate the attacker to plant a payload (e.g., Javascript code) *before* the time-of-archive of the victim page. At the time-of-access, attacks served from an archived version of an attacker's page are passive, whereas attacks served from the attacker's server in the live web are active.

archive itself — a **passive attack** — or from an attacker's live web server — an **active attack**. In a passive attack, the attacker is not actively involved at time-of-access. Specifically, Attacks #1 and #3, which both use archive-escapes, are active attacks, since the attacker's server is the destination of the archive-escape. By contrast, Attacks #2 and #4 deliver payloads the attacker has placed in the archive, and which are delivered to the client by the archive front-end.

**Some Attacks Require Foresight.** Some attacks require *foresight* on the part of the attacker. By foresight, we mean that the attacker must define the attack payload (e.g., the Javascript code to run on the snapshot when viewed by a client) at the time-of-publication of the victim page. Specifically, attacks based on origin-escapes (Attacks #2 and #3) require the attacker to plant malicious code inside an `<iframe>` on the victim page. Attacks which do not require foresight (Attacks #1 and #4) allowing the attacker to choose a payload at any time, including after time-of-archive. For example, in Attack #1, the attacker can even change this payload over time (whereas once an anachronism has been injected in Attack #4, that payload is fixed).

**Partial vs. Full Control.** For all attacks, vulnerabilities may permit either *partial-control* or *complete-control* attacks, depending on the type of resource the attacker controls in the specific instance of the attack. If an attacker controls static resources like text or images, the attacker can only changes those particular elements (partial-control). If an attacker controls

client-side code, such as Javascript or a CSS stylesheet, the attacker can leverage that code for complete-control, gaining the ability to modify *any* part of the client's view of the snapshot, such as its text, styling, images, layout, client-side dynamic behavior, and so on. We explore the prevalence of partial-control and complete-control attacks in the Section 3.6.

## *3.6 Measuring Prevalence of Archive Vulnerabilities*

### *3.6.1 Measurement Methods*

**Measurement Tool.** The authors of [14] provided us with TrackingExcavator, their archival measurement tool. TrackingExcavator is a Chrome extension which automatically visits an "Input Set" of URLs, locates them in the Wayback Machine at a requested timestamp, and collects event traces as it loads and renders those URLs. These event traces include events for all HTTP requests the browser makes, which we use to locate vulnerabilities to our attacks.

**Our Datasets.** Our measurements include measurement traces from three sets of URLs:

For the **Top 500**, we downloaded the publicly available traces from [14]. [4] For the **Top Million**, we used historical versions of the Alexa Top Million CSV file for the years from 2010-2017, which we located in the Wayback Machine [9]. We sampled every thousandth site from those Top Million lists, such that we visited sites with popularity rank 1, 1001, 2001, ..., etc., similar to papers that have sampled from the Top Million [147]. These traces cover a different (but sometimes overlapping) set of URLs in each timestamp year, with a trace for each site's snapshot once for each year in which it appeared in the Top 500 or our Top Million sample.

For our **Legal URL** dataset, we searched Westlaw and LexisNexis for court decisions, court filings, and federal agency administrative decisions which contained the phrase "web.archive.org" [11, 10]. We found that both legal databases contained substantially similar results, and so used only the results from Westlaw. We then located Wayback Machine URLs cited in these materials, collecting separate lists of URLs for each category of legal proceeding (court decisions,

[4] Available at `https://trackingexcavator.cs.washington.edu/`, Accessed 2017-03-30.

court filings, administrative decisions). These include 119 URLs cited in 101 court decisions, 255 URLs cited in 302 appellate briefs, 159 URLs cited in 217 expert material documents, and 307 URLs cited in 371 administrative decisions. [5] We collected traces of the snapshots at the exact URLs cited in the legal materials.

**Measurement Parameters.** We crawled the archive from Amazon EC2 t2.large instances, rendering Chrome (running TrackingExcavator) headlessly inside a virtual frame buffer. We opened 3 tabs at once, one tab per snapshot, and remained on each snapshot for 40 seconds, which [14] found is a sufficient for snapshots to complete loading in the browser. We set TrackingExcavator to block (but still record) archive-escape requests, in order to prevent contaminating our view of the archive with live data. This means we undercount overall archive-escapes that would be seen by an ordinary browser (since we miss archive-escapes caused by other archive-escapes), making our numbers a conservative lower-bound on the archive-escapes a client will encounter in the wild.

### *3.6.2 How Often Are Archived Sites Vulnerable?*

Figure 3.3 depicts the prevalence of all types of vulnerabilities to our attacks in the top panel, and the prevalence of vulnerabilities which allow the most powerful attacks (complete-control without foresight) in the bottom panel. This figure depicts only data from the Top 500 — the trends we found in the Top Million were similar.

**Three-Fourths of Archived Sites Are Vulnerable.** Considering the union of the top sites across all years, we studied 2692 distinct sites from the Top 500 and 7000 distinct sites in the Top Million. We found found that 73% of those Top 500 sites and 80% of those Top Million domains were vulnerable to one of our attacks, either now (for Archive-Escape or Anachronism-Injection vulnerabilities, which do not require foresight) or at time-of-archive (for Same-Origin Escape vulnerable snapshots, which do require foresight).

Recall that for each vulnerable snapshot, there is a limited set of domains which are

[5] In an administrative decision, a U.S. federal agency resolves lawsuit-like cases related to the agency's jurisdiction. They may replace or precede normal lawsuits.

capable of exploiting that vulnerability (e.g., the destination domain of an archive-escape vulnerability, or the owner of the domain of a missing resource). That is, not *anyone* can mount these attacks — only attackers who own or are able to acquire these domains. We consider the number of unowned domains (accessible to anyone) later in this section.

As shown in Figure 3.3, these vulnerabilities are widespread and varied in type, endangering client views of a large fraction of archived sites. Archives and their users should take care to ensure they put appropriate levels of trust in archival data, given the frequency with which they are vulnerable to manipulation.

**Sites Are Vulnerable To Strong Attacks.** While the top of Figure 3.3 considers all of our attacks, the bottom panel considers a particularly strong, category of attacks: Archive-Escape (#1) and Anachronism Injection (#4) vulnerabilities which enable complete-control. Even vulnerabilities to this strong class of attacks are quite common in the archive: 38% of Top 500 domains and 65% of Top Million domains are vulnerable.

**Prevalence of Some Vulnerabilities Has Changed Over Time.** The prevalence of our vulnerabilities varies over the age of snapshots in the archive. For example, more recently captured snapshots are dramatically more likely to be vulnerable to archive-escape abuse. For example, in both the Top 500 and Top Million, the fraction of snapshot domains vulnerable to archive-escape abuse increased from 22% to nearly 80% over the period from 2007 to the present day. We believe that this trend is due to the increasing complexity of sites over the history of the web, since URL rewriting failures, which cause archive-escapes, often result in client-side dynamic behaviors in sites. As sites have grown more complex with more client-side dynamic behaviors, so have the prevalence of archive-escapes and the vulnerabilities that they cause.

**Snapshot Domains Remain Vulnerable Over Archival Time.** The series of snapshots of a site in the archive may span years or decades, as a site ages. We find that not only are individual snapshots often vulnerable (Figure 3.3), but also that many of the websites we studied remained vulnerable over long periods of time. Figure 3.4 shows the number

| Potential Attacker | Number of Possible Victims |
|---|---|
| google-analytics.com | 108 |
| googletagservices.com | 78 |
| facebook.net | 67 |
| googletagmanager.com | 66 |
| doubleclick.net | 59 |
| gstatic.com | 56 |
| criteo.com | 27 |
| amazon-adsystem.com | 22 |
| newrelic.com | 22 |
| cloudfront.net | 21 |

Table 3.2: The third-party attacker domains capable of attacking the most snapshot domains.

of vulnerable domains in each year which were also vulnerable in the previous year. For example, of the snapshot domains which were vulnerable to Archive-Escape Abuse in 2016, about 80% of them were also vulnerable in 2015.

This type of continuous vulnerability suggests that the appearance of vulnerabilities in these sites may be due to structural elements of the way the sites are designed and published, such as publishers' choices to embed third-parties, to use client-side dynamic behavior, and to include third-party Javascript libraries. This implies both that changes in the architecture of these sites might alleviate these vulnerabilities, but also that they are unlikely to go away on their own, especially as many of the more complex aspects of the modern web may lead directly to some of our attacks.

We note that continuous vulnerability of a website may be valuable to attackers who need to modify the appearance of a particular snapshot of a website for their goals. If a large fraction of the snapshots of a website are vulnerable over time, the chances are much greater that an attacker will be able to exploit the particular snapshots needed for their goals.

### *3.6.3 How Many Potential Attackers Are There?*

**Some Potential Attackers Have the Ability to Compromise Many Domains' Snapshots.** Recall that potential attackers are those who own, or can obtain, the domains asso-

ciated with vulnerabilities. There are a total of 2077 Attack #1/#4 attackers over the 2692 sites in our Top 500 dataset (3298 attackers over 7000 sites in the Top Million). Many of these attackers are quite limited in the targets they can attack, with just over half of attackers in the Top 500 only able to attack a single, particular snapshot domain (40% in the Top Million). However, attackers with more widespread opportunities exist. Table 3.2 shows the individual third-party domains which could launch Attacks #1 or #4 against the *most* snapshot domains. Many of these domains are third-party domains which appear as across a large number of sites, such as advertising and analytics networks, social network widgets, and content distribution services. We do not expect any of these companies to maliciously modify the archive; rather, we list them to characterize the types of modern web practices which so frequently lead to our vulnerabilities.

**First vs. Third Party Attackers.** While Same-Origin Escape based attacks (#2 and #3) can only be executed by a third-party domain, both Archive-Escape Abuse and Anachronism Injection attacks (#1 and #4) can be performed by both first- and third-parties. Both of these types of attackers are interesting, although they represent significantly motivated attackers. The first-party is usually the original publisher of the information in the snapshot, and so a first-party attacker is changing content they published, while a third-party attacker is generally changing content which was originally created and published by the first-party. While both first- and third-parties are potentially interesting attackers, we note that individual site owners may be more alarmed by the potential for third-parties to modify their snapshots.

Over the existence of the archive, third-party attackers have become much more common for archive-escape vulnerabilities, to the point that nearly every (97%) recent snapshot with an archive-escape vulnerability includes at least one to with a third-party destination, up from 60% since 2007-timestamp snapshots. We hypothesize that this trend is caused by the combined trends in the modern web of increasing complexity and increasing inclusion of third-parties. By contrast, third-party missing resources have become less common over time. They made up nearly all missing resource vulnerabilities in 1996 (98%), and only

about 40% in 2016.

**Unowned Attack Domains.** Our vulnerabilities enable attacks by particular domains on the Internet, but the ownership of that domain may shift over time. Indeed, attacker domains are sometimes completely unowned. Aggregating across our datasets, we found 23 archive-escape destination domains and 60 never-archived resource domains which were unowned as of Spring 2017. These domains can be purchased by anyone to launch an attack on their vulnerable sites. This is how we performed our proof-of-concept attack (Figure 3.1). We found no unowned attack domains in our legal dataset.

### *3.6.4 Measurements of URLs Used in Court Proceedings*

We now analyze our dataset of the archive.org URLs used in court proceedings. Recall from Section 3.6.1 that this dataset consists of 840 URLs from 991 legal documents. Because they have been cited in court proceedings, the accuracy of these archived pages is critical — or, conversely, the motivation clearly exists for a potential attacker to manipulate one of these snapshots to influence legal proceedings.

In this section, we thus investigate the prevalence of vulnerabilities in these snapshots. We stress that the presence of a vulnerability does *not* imply that an attack actually occurred. Indeed, evaluating the question of whether an attack occurred is challenging, since, for most attacks, they can be temporarily enabled and then disabled. Instead, our goal is to survey the prevalence of these vulnerabilities in specific archives that *have* been used in legal proceedings in the past, to serve as a note of caution for the use of archived URLs in future proceedings.

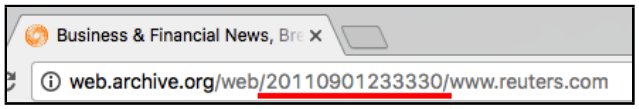
For these legally referenced snapshots, we considered only Attacks #1 and #4, which do not require foresight, and thus could be mounted after the fact, at the time of legal proceedings. 57 were vulnerable to Attack #1, and 37 of those were complete-control vulnerabilities. However, *none* contained never-archive resources, which is quite unlike the archive at large, which commonly contains never-archived resource vulnerabilities (Figure 3.3). We hypothesize that URLs cited in legal proceedings may be of higher quality since they were curated

by experts deciding which URLs to cite.

If these vulnerabilities had been exploited at the time of these legal cases, they could have given an attacker the ability to hide or plant evidence. Again, we stress that we have no reason to believe that any of these vulnerabilities were exploited at the time of the relevant court proceedings, but emphasize that future use of archived URLs in legal or other similar matters must be treated with caution.

## *3.7 Defenses*

In this section, we explore the space of possible defenses against our attacks, including defenses which *detect* or *block* our attacks. As an overall defensive goal, we aim to allow users of archives to have more confidence in their understanding of the web of the past.




(a) Above, the snapshot URL for our demonstration attack, a capture of the Reuters homepage from the timestamp 20110901233330 (1 September 2011, at 23:33:00).


(b) Above, the original news story from the page, as preserved in the snapshot URL above: a political opinion piece, illustrated with a picture of President Barack Obama. Accessed 15 May 2017.


(c) Above, we used an Archive-Escape Abuse attack (Section 3.5.1) to replace the above article with incorrect content, so that clients would see CCS 2017's cover image and a 6-year-early prediction of CCS 2017's host city rather than the correct election opinion piece.

Figure 3.1: We enabled this attack only or the purposes of obtaining this demonstration screenshot, and disabled the attack after determining that it worked.

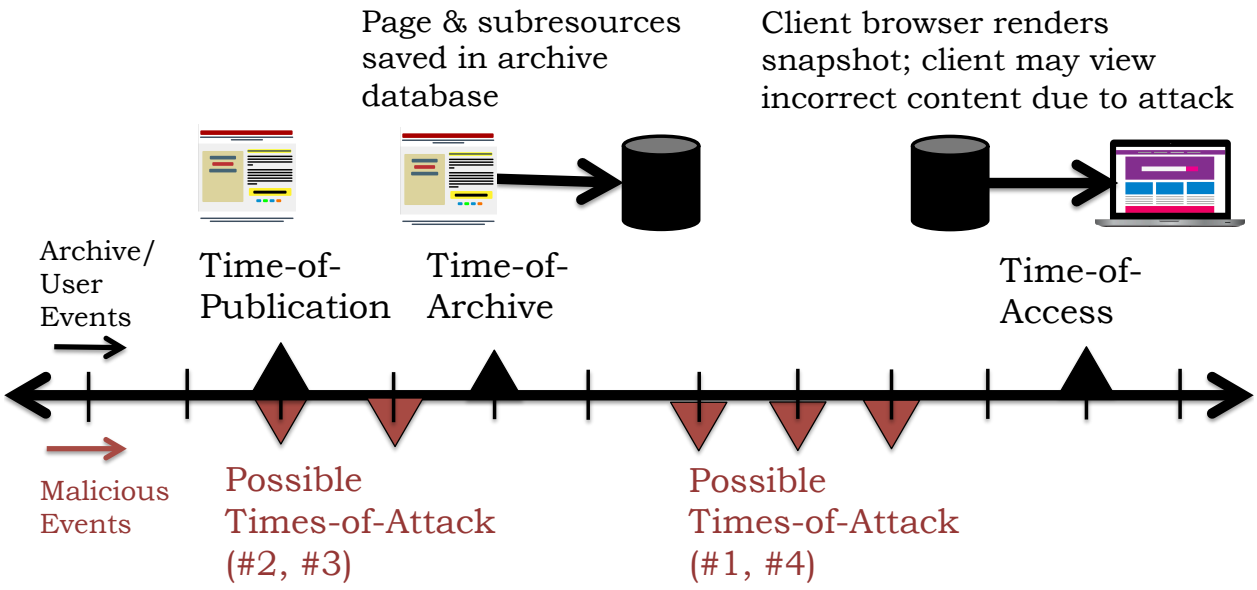



Figure 3.2: A timeline depicting the (1) lifecycle of archive snapshots (top of figure) and (2) events that make up attacks against the integrity of those snapshots (bottom). The left-hand possible Times-of-Attack, before Time-of-Archive, correspond to Attacks #2 and #3, which require attacker foresight. The right-hand possible Time-of-Attack is after Time-of-Archive (but still before Time-of-Access), for Attacks #1 and #4, which do not require attacker foresight. Attacks are described in detail in Section 3.5.

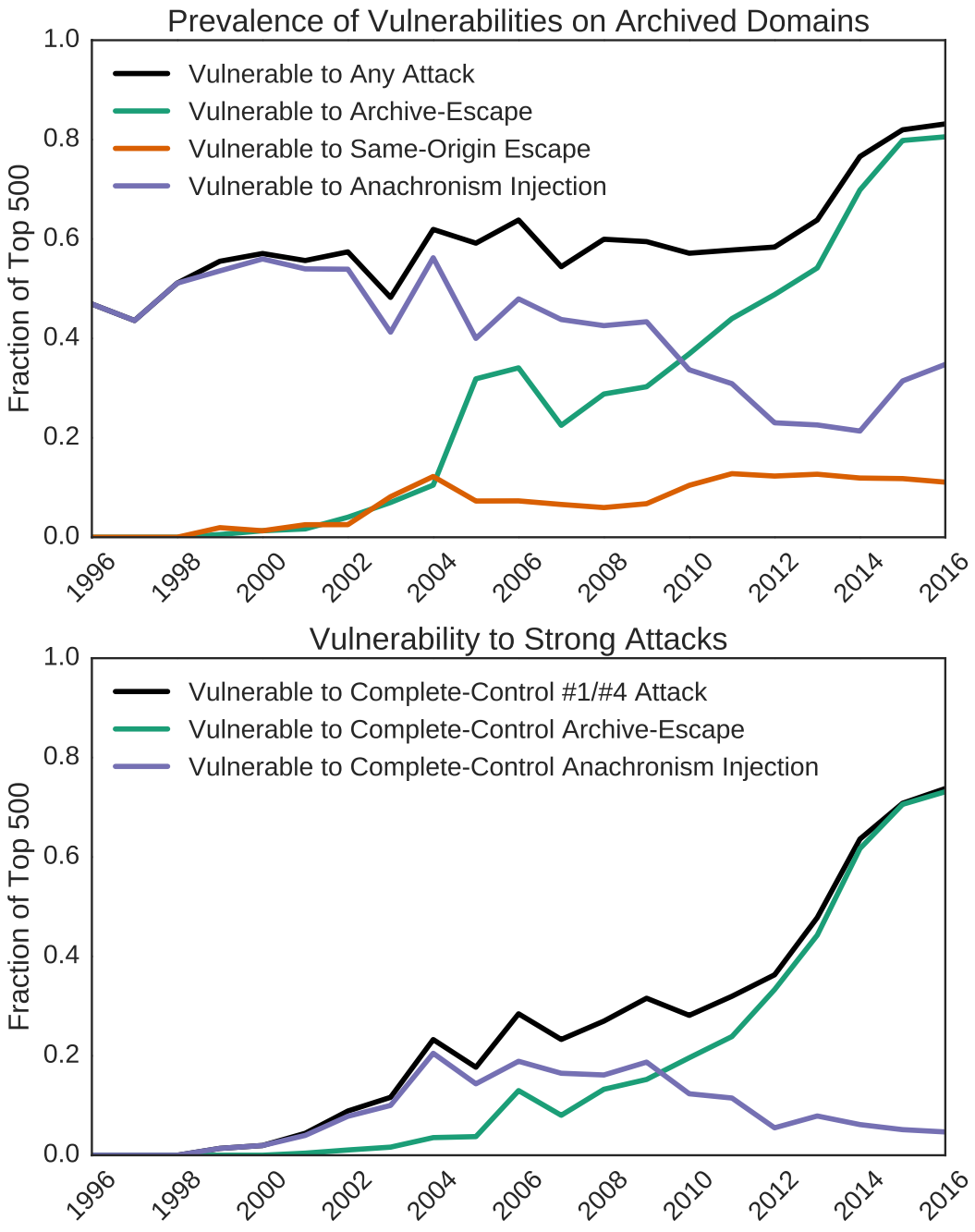



Figure 3.3: Top: The prevalence of vulnerabilities to our attacks across the Top 500 sites. Bottom: The prevalence of vulnerabilities to the particularly strong class of attacks which provide complete-control without foresight (Attacks #1 and #4 with script/stylesheet as vulnerable resource). Not shown: Our Top Million dataset shows very similar trends to the Top 500.

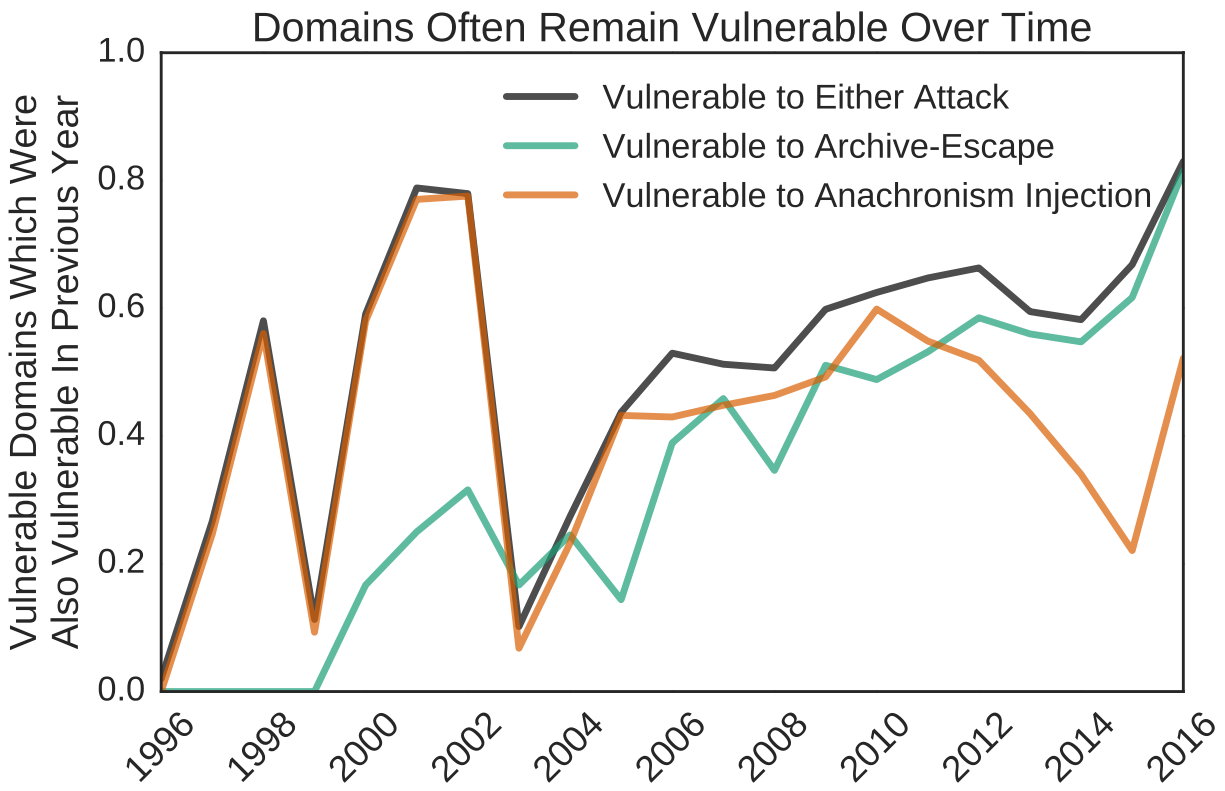



Figure 3.4: The increasing tendency of snapshots to remain vulnerable to our attacks across subsequent years. This figure represents the number of snapshot domains in each year whose snapshot from the previous year was also vulnerable to the given attack(s).

| Defense | Goals | | Who Deploys? | When? |
|---|---|---|---|---|
| | Prevent | Detect | | |
| Opt-Out of Archives | ✓ | | Website Owner | Any Time |
| Avoid Dynamically Generated URLs | ✓ | | Website Owner | Time-of-Publication |
| Actively Archive Subresources | ✓ | | Website Owner | Time-of-Archive |
| Modify Archived Javascript to Avoid Escapes | ✓ | ✓ | Archive | Any time |
| Serve Distinct Archived Domains from Distinct Subdomains | ✓ | | Archive | Any time |
| Escape-/Anachronism- Blocking Browser Extension | ✓ | | End-user | Time-of-Access |
| Escape-/Anachronism- Highlighting Browser Extension | | ✓ | End-user | Time-of-Access |

Table 3.3: A summary of the defenses we explore.

We organize our defenses first by **who** deploys them: website publishers, archives, or clients, and categorize them additionally by **when** they can be deployed (i.e., whether they work retrospectively, after time-of-attack). Table 3.3 summarizes these defenses, and we discuss them in detail below. We also we present the implementation of ArchiveWatcher, a browser extension which detects and blocks archive-escapes and anachronisms.

### *3.7.1 Defenses Deployed by Website Publishers*

We begin with defenses website publishers can deploy to protect snapshots of their won websites. These defenses work for all clients, but must be separately deployed by each website, and some are not retroactive, since publishers cannot modify previously archived data. First-party attackers, may avoid deploying these defenses to retain editorial power over their site's past.

#### *Opt-Out of Archives*

Websites can opt-out of being preserved in the Wayback Machine, sidestepping the possibility of archival vulnerabilities. The Wayback Machine has long respected website publishers' opt-out preferences in two ways: manual requests, and the use of `robots.txt` policy files. By opting out of preservation entirely, a site would avoid having snapshots which could be manipulated, preventing all attacks in this chapter.

The downside to this defense is that the relevant sites are not archived or available for the public to browse in the archive, eliminating all the social and cultural benefits the archive brings. This defense throws the baby out with the bathwater. Some sites may also not be legally permitted opt-out, such as government sites with archival requirements. Additionally, this defense may soon become much less viable: Wayback Machine has expressed, in a recent blog post, an intent to give less weight to `robots.txt` files, saying that as of April 2017 it now ignores `robots.txt` on U.S. government and military websites and is "looking to do this more broadly." [8]

*Avoid Dynamically Generated URLs to Avoid Archive-Escapes*

Website publishers can reduce the incidence of archive-escapes by designing their websites to use fewer dynamically generated URLs, since these are a common cause of archive-escapes.

This approach has two major weaknesses. The first is that dynamic behavior and URLs are a common, valuable feature of the modern web, and asking engineers to do without them could be inconvenient and expensive. Second, this defense cannot protect against archived-escapes caused by third-party content, such as third-party Javascript libraries, which are commonly used and whose behavior is not fully under the control of the publisher.

*Actively Archive Subresources*

In Anachronism Injection, the attacker wants to replace a subresource which has never been archived with a malicious payload. One way to defend against this attack is to preemptively replace missing subresources with benign resources, plugging the vulnerability. Though anyone can use the "Save Page Now" feature to plug vulnerabilities — the same feature attackers use to archive their payloads — website publishers wishing to defend their pages in the archive likely have the greatest incentive to do so. However, if no benign resource is published at the URL, the defense will not work. The non-malicious content could be the correct content which was originally present at the URL, an empty response, or even a 404 Not Found response. In all these cases the archive will record the given response as the only capture of the resource and serve it, causing no harm, as the nearest-neighbor to the vulnerable reference.

The most significant limitation of this defense is that only the potential attacker can publish a benign resource to be archived — the permission to enact this defense lies with the potential attacker. While anyone can ask to "Save Page Now" for any URL, this process only works for resources where the server responds to the crawler's response with *some* response, even if it is simply a 404 error. Thus attackers who wish to ensure against malice by themselves in the future, or by later owners of their domain, can use this defense, but it

will be ineffective when the first-party wants to launch an attack.

### *3.7.2 Defenses Deployed by Web Archives*

Defenses deployed by archives have the potential to be quite powerful, since archives can change the data they store in their database (as they do with URL rewriting) and the data they collect in the future, to provide both forward-looking and retroactive defense which protect the views of all clients.

#### *Modify/Analyze Javascript to Prevent Escapes*

In this defense, the archive would statically and dynamically analyze Javascript code it captures in order to identify scripts might cause archive-escapes. The archive would then rewrite or wrap these scripts, replacing the original script with a version that performs the same operations but avoids generating archive-escapes. For example, such a defense might hook calls to browser APIs which generate HTTP requests, interposing on them to rewrite URL arguments to ensure they do not point outside the archive.

This solution is complex, and its implementation might involve many engineering hours. Additionally, executing the defense on each archived resource at time-of-archive might be computationally expensive. However, if successful, this defense might permit the Wayback Machine's URL rewriting to be much more pervasive, applying even to client-side dynamically generated URLs, the main source of vulnerabilities that we identify in the archive today.

#### *Serve Distinct Archived Domains from Distinct Subdomains*

Archives could defend against Same-Origin Escapes by serving content from distinct subdomains, each of which corresponds to the live domain from which that content was originally published. For example, an archive might choose to serve captures of `example.com/script.js` from the subdomain `http://example.com.web.archive.org/` instead of from `http://web.archive.org`. Since the Same-Origin Policy considers subdomains as distinct

domains, this would cause client browsers to provide the same isolation in the archival context as they do in the live context, preserving the same trust model across both live and archival executions of the page. We recommend that archives consider implementing this defense.

### *3.7.3 Defenses Deployed by Clients*

Finally, we discuss defenses deployed inside the client's browser. Individual clients can unilaterally deploy these defenses, giving them high value today. For example, experts in legal cases might use these defenses to provide more trustworthy testimony. These defenses are limited by the fact that each client must separately install the defense, but they do apply to all snapshots in the archive.

#### *Browser Extensions to Block/Highlight Escapes and Anachronisms*

This defense interposes on and blocks Archive-Escape and Anachronistic requests made for subresources while browsing the archive. It prevents Archive-Escape Abuse by blocking all HTTP requests from a snapshot which leave the archive. Since the distinction between archive-escapes and archival requests is cut and dry (distinguishable by the destination domain of the request), such a defense should be highly effective against Archive-Escape Abuse.

This defense protects against Anachronism Injection not by preventing the payload from being stored in the archive (as does the *Actively Archive Subresources* defense, above), but by blocking the anachronistic request which delivers that payload to the client. It does so by blocking anachronistic requests — those requests for archival resources which have timestamps far from the timestamp of the enclosing page. This involves an inherent tradeoff, in which the defense or its user must define *how* anachronistic a resource must be to be blocked. In the most extreme case, only resources with timestamp exactly equal to the snapshot's timestamp can be loaded, leading to complete blocking of the vulnerability, but also preventing many legitimate resources from being loaded, leading to a less complete picture of the past web.

This defense can also (or instead) visibly highlight, log, or summarize archive-escapes and anachronistic requests and the visible page elements which correspond to them. Such a feature can help a human expert to better judge the accuracy of a snapshot. ArchiveWatcher, described in more detail below (Section 3.7.4), is an example of this type of defense.

### *3.7.4 ArchiveWatcher: An End-User Defense*

We prototyped ArchiveWatcher, a client-deployed defense consisting of a browser extension which detects and blocks archive-escape and anachronistic request vulnerabilities. ArchiveWatcher is implemented as a lightweight Chrome Extension which interposes on requests made for resources while browsing snapshots `https://web.archive.org/web`. It is written in 6000 lines of Javascript, CSS, and HTML, and we intend to make it publicly available on the Chrome Web Store upon publication of the conference paper of the work described in this chapter.

As described above in Section 3.7.3, ArchiveWatcher blocks requests for archive-escapes and anachronisms, and has a configurable time window for defining anachronistic timestamps. It can display to the user a summary of the requests it has detected and blocked on the current snapshot. We anticipate that ArchiveWatcher or a similar defense could aid technical experts assessing the veracity of archival snapshots.

## *3.8 Conclusion*

In this chapter, we have explored the space of attacks which can rewrite history — i.e., attacks that can manipulate how clients see archived websites, focusing on the Wayback Machine. Though it is known that the archive contains accidental inaccuracies, to our knowledge, we are the first to explore how an attacker might introduce *intentional* errors. We identified and explored several vulnerabilities in how the Wayback Machine archives and serves snapshots of websites, and we developed four attacks that leverage these vulnerabilities. We demonstrated proof-of-concept attacks on the Wayback Machine, showing that we were able to manipulate client views of snapshots without compromising the archive's or any other servers. We then

quantified the prevalence of these types of vulnerabilities, finding that over 70% of the sites we investigated are vulnerable to this type of manipulation by *some* attacker.

The web is important to our modern society, making web archives a critical source of socially important information, from journalism to legal proceedings. This work suggests the importance for website publishers, archive designers, and end users to take steps to prevent or detect intentional manipulation.

# Chapter 4

# ANALYZING THE USE OF QUICK RESPONSE CODES IN THE WILD

In this final contribution chapter, I focus on the sister technology of QR codes, considering their use across space and time and with an eye toward their relationship to the web. These measurements are, to our knowledge, the first large-scale academic analysis of how QR codes are used in the wild, and the facts and lessons I drive shed light on the ways that emerging sister technologies which embed information in the environment might interact with established core technologies like the web.

The work described in this chapter previously appeared in a 2015 paper [109], and citations to this work should refer to that conference publication.

## *4.1 Introduction*

With the growing prevalence of smartphones and other mobile devices, Quick Response (QR) codes have become a convenient way to quickly communicate a small amount of information, such as a URL, to a user's device. Figure 4.1 shows a sample QR code; Figures 4.2 and 4.3 show examples in real-world contexts. To read these codes, users typically install third-party QR code scanning applications onto their mobile devices. The number and popularity of such applications speaks to the popularity of QR codes themselves. For example, the most popular QR and barcode scanning application for Android boasts over 100 million downloads (as of November 2014, according to `http://appbrain.com`).

QR codes are among the most prevalent technologies *bridging the physical and digital worlds*, raising unique opportunities and challenges. QR codes are often associated with physical objects. When a user scans a QR code with a mobile device, that mobile device

Figure 4.1: Sample QR code.

may perform some follow-on digital action, such as visiting a website (Figure 4.2) or pairing accounts (Figure 4.3). Anecdotally, QR codes are used for marketing purposes or to provide pointers to additional information about a physical location or object (e.g., in a museum). QR codes are also used for a growing number of security-sensitive operations, such as authentication, device pairing, and connecting to password-protected WiFi networks. The prevalence and utility of QR codes is likely to increase with the growing adoption of wearable devices such as Google Glass, where text or other traditional forms of input may be cumbersome. For example, Google Glass already utilizes QR codes to connect to password-protected WiFi networks [60]. The research community has also turned to QR codes as a mechanism for linking physical spaces with digital information or computation (e.g., [24, 150, 146, 37, 21, 87, 148, 123, 71, 153, 110, 38] — see Section 4.2 for a discussion).

Though QR and barcode scanning applications and anecdotes about their uses abound, to our knowledge there has been no systematic, in-depth study of their use in the wild. Designers of QR code-based systems are thus forced to rely on speculation, or their own measurements, of the QR code ecosystem. To fill this gap, we study the use of QR codes in the wild. We leverage a unique and powerful data set: approximately 87 million scans of QR and barcodes made using Scan (`https://scan.me/`), a popular scanning application with an install base of over 10 million devices.[1] This dataset allows us to examine both the types of QR codes that exist in the wild, and the frequency with which individuals users interact with these codes. We find, for example, that approximately 75% of QR and barcode

[1]These scans were logged in accordance with Scan's terms of service and privacy policy, and our use of the dataset was approved by our institutions' IRB boards.


Figure 4.2: A QR code in a museum, encoding the URL for a video related to the exhibit. Image source: `https://www.flickr.com/photos/balboaparkonline/`


Figure 4.3: A QR code used to pair a user's YouTube account with the YouTube app on their Xbox. Image source: `http://instagram.com/p/icywcsKZbo/`

scans in our dataset lead to web URLs, and that the set of popular websites found in QR codes differs significantly from the set of websites popular on the web in general. We also find that 25% of scans represent other use cases, and we investigate these varied applications (e.g., phone numbers, Bitcoin payments, and two-factor authentication). We also observe examples of malicious uses of QR codes, including links to Android applications containing malware. These cases of misuse are rare, but their presence in our dataset suggests that users may encounter them in the wild.

In this chapter, we analyze the use and misuse of QR codes, and we develop an informed understanding of the types of codes that are created and that are encountered by users in practice. Our contributions include:

- We conduct the first (to our knowledge) large-scale academic analysis of QR and barcode use in the wild, using a dataset of 87 million scans. We present general trends

about the scans, codes, and devices present in our data.

- We investigate specific use cases for QR codes, including a deep dive into the content of popular codes in our dataset, a comparison between frequently and infrequently scanned codes, and an exploration of the varied use cases present in our data (e.g., cryptographic currencies, device pairing, one-time passwords). We also investigate several potential vectors for malicious QR codes, including malware or phishing URLs and links to malicious Android applications.
- From these findings, we distill a set of lessons and recommendations for QR and barcode scanning application designers as well as for future systems that rely on QR codes or similar techniques to communicate between objects.

We now provide additional background and discuss related work in Section 4.2 before describing our dataset in more detail in Section 4.3. We then present general analyses of the dataset (Section 4.4), and then an analysis of different use cases that manifest in the dataset (Section 4.5). We discuss the implications of our findings and avenues for future work in Section 4.6 and then conclude in Section 4.7.

## *4.2 Background and Related Work*

One- and two-dimensional barcodes have become popular as a convenient way to quickly communicate a small amount of information, such as a URL, to a user's device. Figures 4.2 and 4.3 show examples of Quick Response (QR) codes, a common type of two-dimensional barcode. The prevalence of QR codes has risen alongside the popularity of smartphones [115], with one study showing that European usage of QR codes doubled between 2011 and 2012 [32]. To read these codes, users can install on their devices a variety of third-party QR and/or barcode reading applications, such as the popular ZXing Barcode Scanner, which boasts over 100 million downloads on Android, or Scan (`https://scan.me/`), which has been downloaded over 10 million times for Android.[2]

[2]Download numbers for Android, according to `http://appbrain.com` in November 2014. The total number of downloads for all platforms is higher.

**Research applications.** QR and barcodes have been applied in a variety of research contexts. Several efforts leverage the ability of these codes to bridge the physical and digital worlds [24, 150], such as for indoor navigation [146], grocery bargain hunting [37], accessibility [21], and to aid augmented reality applications [87]. Additionally, many security and privacy related uses of QR and barcodes have been proposed, including for communicating privacy policies [148, 29], device-to-device authentication [123], web authentication [71], encryption or verification of real-world paper content [153, 110], and as tattoos of medical device passwords [38].

**Security issues.** Other researchers have explored the security challenges raised by QR and barcodes, including attacks enabled by ambiguous decoding protocols [36], a study of people's susceptibility to QR code based phishing attacks [168], the potential use of QR codes to spread malware and phishing URLs [176], and other attacks [91]. While these attacks are all technically possible, how prevalent are they in practice? We investigate several types of potentially malicious QR codes in this work, including malicious URLs.

**Commercial use.** Commercially, QR codes are frequently used for marketing purposes [79, 30], but they have also been applied in various security-sensitive contexts, such as authentication or device pairing. For example, Google experimented with QR codes for passwordless login [133], and Google Glass uses QR codes to connect to password-protected WiFi networks [60]. Indeed, the prevalence and utility of QR codes is likely to increase with the growing adoption of wearable, camera-enabled devices like Google Glass.

**Missing knowledge about real-world use.** Though QR and barcodes have been frequently applied in a diversity of research and commercial applications, to the best of our knowledge no large-scale academic study has been conducted of the use of such codes in the wild. Thus, we have little concrete knowledge about the prevalence of the various applications and attacks described above. We also have little concrete knowledge about the behavior of real users who may encounter QR codes. We aim to close that gap in this chapter, leveraging our unique dataset of 87 million scans from a popular QR and barcode scanning

Table 4.1: The schema of our dataset. Device UUIDs are random and each corresponds to a single device which has installed the Scan app.

| *Column* | *Example* |
|---|---|
| Barcode Type | e.g., QR, UPC, etc. |
| Contents | URI or other data |
| Location | Lat/lon coordinates |
| City | City |
| Region | e.g., state, province, etc. |
| Country | Country |
| Platform/Version | Mobile OS and version |
| Device Type | Phone make and model |
| Device ID | UUID |
| Timestamp | Date and time |
| Scan Source | Camera/History/Gallery |

application. In the next section, we detail our dataset.

### *4.3 The Dataset*

**Origin and Scope.** We obtained a log of scans performed by users of Scan (`https://scan.me`), a popular barcode and QR code scanning application for Android, iOS, and Windows Phone. The log includes scans made by real users using the software over a 10 month period from May 2013 to March 2014, including 87,647,504 scans of 18,763,779 distinct barcodes by 15,484,921 distinct devices in 241 countries.

**Schema.** Table 4.1 describes the full schema of the dataset. It includes the location and time of each scan as well as an anonymized UUID distinguishing devices which have installed the app. We note that devices are not one-to-one with users, since a user may have multiple devices at once or over time, and a single device may have multiple users. Hence, for expository purposes, when we refer to "users" in this chapter, we are often referring to devices.

**Human Subjects.** The dataset was collected for academic research purposes in accordance

with Scan's terms of service and privacy policy (e.g., for Android: `http://scan.me/mobile/apps/scan/android/legal/eula`). We also received IRB approval for this study from the IRBs of the University of Washington and Brigham Young University.

When not stored at Scan according to Scan's policies, we store the data in an encrypted form and performed all of our analyses on machines that we physically control. Though the dataset contains some personally identifiable information (e.g., names and phone numbers in QR codes encoding business cards), we report only aggregate or anonymized data in this chapter.

**Definitions.** Throughout the chapter we refer to a "code" as a distinct piece of data contained in a barcode or QR code. For example, Alice and Bob might both embed the url `http://example.com` in a QR code. If a user scans Alice's code and another user scans Bob's code, we consider the *code* `http://example.com` to be scanned twice.

A scan is the act of a user scanning a code with the app, corresponding to one row in the dataset, with all the fields described in the schema (Table 4.1).

## *4.4 General Analyses*

We begin by presenting an overview of the barcode and QR code scans in our dataset. We analyze the relative popularity of different types of codes, and examine variations over time and geographical region.

### *4.4.1 Basic Distributions*

**Devices.** Our dataset includes 15,484,921 distinct devices, each of which corresponds to a user's mobile device such as a phone or tablet. These devices serve as proxies for users of the app, though we note that devices may not correspond directly to users (e.g., a user may have multiple devices).

We find that a minority of devices account for the majority of scans in our dataset. Specifically, the most prolific 10% of devices performed approximately half of the scans,

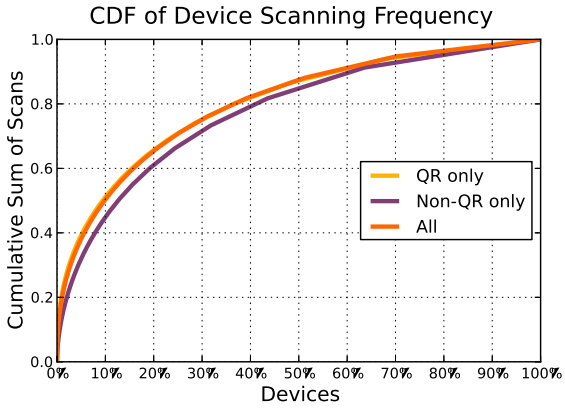



Figure 4.4: The distribution of scans across devices. Devices had nearly identical distributions of scanning for QR codes and all codes combined — the yellow (QR only) and orange (All) lines overlap almost entirely. The right hand side of the figure shows 30% of devices which contributed only one scan each. Heavy hitters on the left: 10% of devices performed half the total scans in the dataset.

while just over 30% (4,667,012) performed only one scan over the 10 months covered by our study. Since the dataset is naturally truncated, users who installed the app near the end of our study period will appear to have very few scans. Thus the proportion of infrequent users is slightly exaggerated.

Figure 4.4 shows the distribution of scans across devices. That figure presents the data over all scans, as well as for only QR code scans and only non-QR code (product barcode) scans. We note that the distribution remains very similar in each of these cases. We speculate that this suggests that the tendency to use a mobile device to scan barcodes is influenced not only by the location and context of barcodes in the environment but also significantly by the individual person's experience and skills — perhaps based upon their affinity to technology. If true, this has implications for adoption of these types of technologies: it may suggest that the uptake of technologies like mobile barcode scanners may depend more on user familiarity or skill than on the ways barcodes are deployed in the environment.

Our results confirm that one- and two-dimensional barcodes can be an effective mechanism for having a physical device influence a digital device for *some* users. For example, 10% of users (accounting for over 1.5 million devices) performed more than 43 million scans

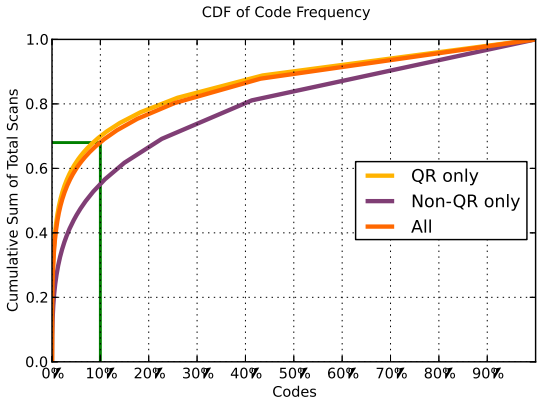



Figure 4.5: The distribution of code popularity. The top 1% of codes were extremely popular: they made up over 42% of total scans. 10% of codes accounted for over 65% of the total scans that occurred, while in the tail, just over half of codes were only ever scanned once. We see similar trends among all codes, QR codes, and non-QR codes.

in our dataset. The existence of these users supports the use of QR codes in emerging technologies and research projects. On the other hand, QR codes do not currently seem to have sufficient appeal for all users (e.g., the 30% of devices with only one scan), thus suggesting that applications that involve a QR-code-based path may not (yet) see much adoption.

**Codes.** The 18.7 million distinct codes seen in the dataset also followed a typical heavy-hitter vs. long-tail distribution, with a small number of codes scanned many times and many codes scanned only once or a few times. Figure 4.5 shows the distribution of popularity amongst codes. The most popular code in the dataset had 244,660 scans (0.28% of all scans) while the top 11 individual codes accounted by themselves for 1% of all scans. (We note that the proportion of unpopular codes is artificially inflated by the fact that codes introduced near the end of our time period will necessarily show a low scan count, even if they were subsequently scanned many times.)

Counter to our initial hypotheses, about half of all codes were only ever scanned once, and only a small fraction of codes reach a large number of people. These popular codes speak to a particular set of uses and user experiences of mobile barcode scanning; we explore

Table 4.2: Distribution of barcode types appearing in all scans in our dataset. Note that this table counts the appearance of code types in scans, i.e., codes that were scanned more than once are counted once per scan.

| *Barcode Type* | *Count* |
|---|---|
| QR | 76,304,319 (87.06%) |
| EAN-13 | 6,554,534 (7.48%) |
| UPC-A | 3,701,269 (4.22%) |
| EAN-8 | 687,889 (0.78%) |
| UPC-E | 399,493 (0.46%) |
| *Total* | 87,647,504 (100%) |

these popular codes in detail in Section 4.5.1. For example, we observe that many of the most popular codes are web links to sites of corporations, suggesting that heavy-hitters may be primarily scanned due to their presence in successful marketing efforts.

In addition, we find that less popular codes are more likely to correspond to a different set of applications, including interesting emerging applications such as cryptographic coins, business cards or WiFi pairing. These results thus suggest that QR codes are an attractive tool for designers of emerging mobile technologies. We explore these less popular codes in Sections 4.5.2 and 4.5.3.

**Code Types and URI Schemes.** Scan supports scanning both ordinary barcodes as well as QR codes. QR codes dominate usage of the app: about 87% of all 87 million scans were of QR codes, with the remaining 13% divided between different types of one-dimensional EAN and UPC product barcodes. Table 4.2 shows this breakdown.

While one-dimensional product barcodes encode only numbers, QR codes can contain arbitrary text. When this arbitrary text contains something more than a direct web URL, it is often made more useful by structuring it to contain a URI scheme, such as `tel:` for telephone numbers or `mecard:` for business cards. Table 4.3 describes the distribution of URI scheme among the scans in our dataset.

In our dataset, we find that the use of QR codes to encode web URLs dominates: as

Table 4.3: Distribution of URI schemes appearing in all QR code scans in our dataset. This table shows the number of scans of codes encoding actions in various protocols. Percentages are reported out of the 76 million QR code scans, not the total 87 million scans (that include barcode scans).

| *URI Scheme* | *Note* | *Count* |
|---|---|---|
| `http://` | | 58,488,390 (76%) |
| `https://` | | 7,640,420 (10%) |
| `mecard:`/`vcard:` | Business cards | 1,759,773 (2.3%) |
| `market:` | Android app store | 197,407 (0.25%) |
| `smsto:` | Send SMS | 180,752 (0.23%) |
| `otpauth:` | Two-factor auth | 172,091 (0.22%) |
| `wifi:` | Connect to Wifi | 133,963 (0.17%) |
| `tel:` | Phone number | 123,150 (0.16%) |
| `bitcoin:` | Crypto currency | 39,073 (0.05%) |
| `itms-services:` | iOS app store | 30,663 (0.04%) |
| `litecoin:` | Crypto currency | 11,796 (0.01%) |
| `dogecoin:` | Crypto currency | 317 (¡0.01%) |
| Other | | 7,526,524 (9.9%) |
| *Total* | | 76,304,319 (100%) |

reflected in Table 4.3, about 86% of QR code scans (about 75% of all scans, including non-QR code scans) contained a web URL. This suggests that quickly connecting a mobile device to a website is by far the most common use case for QR codes. Of these web URLs, we find that only about 10% specified SSL through the `https:` URI scheme.

Though web URLs dominate the QR code scans in our dataset, we nevertheless observe that 14% of QR code scans (about 25% of all scans, including non-QR code scans) contain something other than a web URL. These 10,175,509 scans represent a non-trivial engagement with a variety of other use cases. For example, we did not initially anticipate the prevalence of some URI schemes, such as `wifi:` and `bitcoin:`. We return to discussing such use cases in Section 4.5. A key takeaway from this general analysis, however, is that our dataset provides strong evidence that QR codes are used for many things besides websites.

**Data Density.** QR codes that encode less information are less visually dense. We investigate the distribution of densities among QR codes in our dataset. We find that on

Table 4.4: Data density of codes of varying frequency of scanning.

| *Popularity* | Bytes/code *Mean* | *Median* |
|---|---|---|
| Infrequent (1-5 scans) | 58.7 | 43 |
| Moderate (6-100 scans) | 49.9 | 32 |
| Frequent (¿100 scans) | 40.9 | 30 |

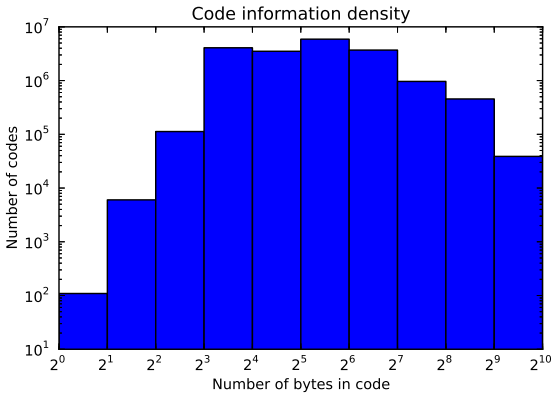



Figure 4.6: Histogram of data density in codes, on a log scale. Codes of length 8-128 are most common.

average, more popular codes encode about 18 fewer bytes of information than unpopular codes. The difference is smaller in the median, with a difference of 13 bytes between popular and unpopular codes.

This information suggests that successful, widely scanned QR codes tend to be shorter. We cannot tease apart from this fact whether people who create popular codes tend to make shorter codes (e.g., they tend to include nothing but a URL or explicitly optimized) or whether shorter codes are more likely to be scanned (e.g., a shorter code is processed by scanning applications more quickly, and therefore is more likely to be scanned before the user gives up in frustration).

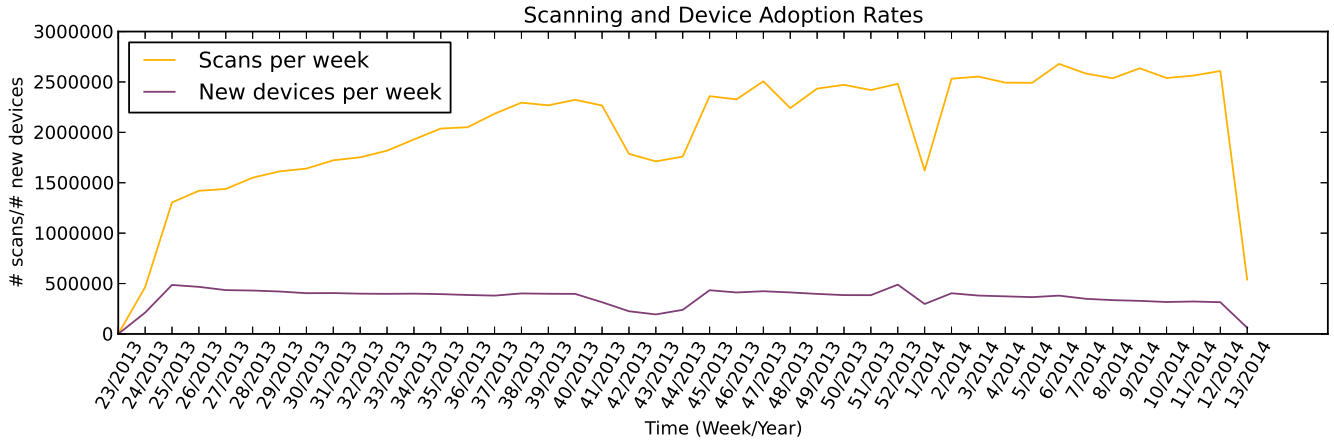



Figure 4.7: Comparisons between scans per week and new devices appearing in the data per week, which is a proxy for user adoption of the app. These two properties are significantly correlated ($R^2 = 0.295$), suggesting that new users of these types of technologies may be a significant driver of the technology's usage in general. Note that the first and last data points are low because our data for those weeks is incomplete; similarly, the last week of 2013 is low because it is not a full seven days.

### *4.4.2 Analyses over Time*

**Usage Trends.** Our above analyses suggest that one- and two-dimensional barcodes are effective means to reach some users. We now ask: how do users engage with QR codes over time? For example, once a user installs the app, do they steadily scan codes over time, or does their usage peak initially due to factors like the novelty of the technology?

To investigate this question, we first examined user adoption rate throughout the time period of the data, looking at the first time each device was seen. Adoption rate remained relatively constant throughout the 10 month span — Figure 4.7 shows the weekly adoption rate by new devices, which hovers below 500,000.

We then examined the number of scans that took place in each week of the studied 10-month time period. The results are also shown in Figure 4.7. The rate fluctuated from about 1.3 million at the start to about 2.67 million in February of 2014, increasing initially and then leveling off around 1.5 million scans per week.

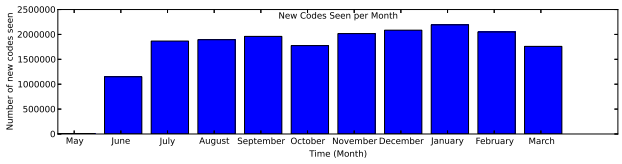



Figure 4.8: Number of new codes seen per month.

Comparing the two lines in Figure 4.7, we observe that although new users appear in the dataset at a regular rate, the number of scans does not increase at the same rate. There are several possible factors that may contribute to this trend besides users whose usage decreases after initial installation and exploration. For example, when a user replaces one device with another, that new device will contribute to the adoption rate but not cause an increased number of scans. Overall, however, this trend suggests that not all users continue to use the app at the same rate after initial installation.

**Codes.** We also explore the appearance of new codes over time. Figure 4.8 shows the rate of the appearance of new codes. Each bar shows the number of codes with unique text content, never seen up to that point in time, that were first scanned during that month. The high rate of appearance of new codes suggests that the ecosystem of QR and barcodes present in the physical world is constantly changing.

Note that the rate of new code appearance is quite similar to the rate of new user adoption. This trend suggests an active interest in the QR code ecosystem for both creating new codes and experimenting with the scanning of codes.

### *4.4.3 Variations by Geographic Region*

Finally, we consider variations by geographic region in our dataset. We use location data reported by devices themselves when they perform a scan.

**Scans.** Our data includes occurring in 241 countries. Of those countries, 19 had at least 1,000,000 scans and 60 had at least 100,000. We note that geographical diversity in scanning might be explained by the popularity of different QR and barcode scanning apps rather than

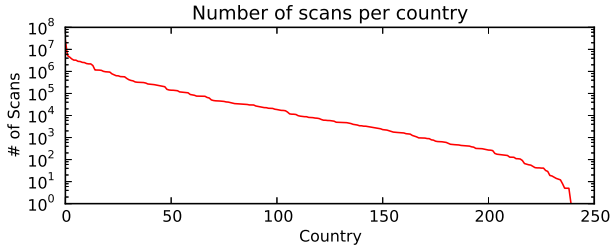



Figure 4.9: The distribution of scans across the 241 countries seen in the dataset. The United States is at the top with an order of magnitude more scans than its nearest competitor, Germany. Note, however, that the United States has a much larger population than Germany.

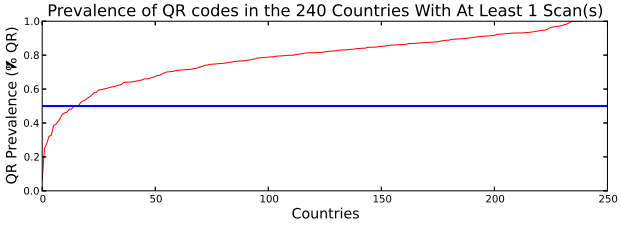



Figure 4.10: Percentage of scans which were of QR codes (as opposed to 1D barcodes) per country for all countries. The blue line indicates the 50% QR mark — countries below the line performed more 1D barcode scans than QR code scans, whereas countries above the line performed predominantly QR code scans. 6 out of 15 of these 1D dominated countries had fewer than 1000 total scans in the database, and none of the 15 had more than 51,000 scans.

the popularity of QR codes themselves. Our vantage with this dataset cannot distinguish such a difference.

Figure 4.9 shows the distribution of scans across countries worldwide, and Table 4.5 shows the number of scans in the most popular 32 cities in the dataset. We find that the top-scanning cities are quite geographically diverse, including cities in the United States, Europe, and Asia. These results suggest that QR codes are a global phenomenon, and not something restricted to a particular region of the world.

**Codes.** Looking across geography, we find that countries vary dramatically in the patterns of their usage of barcodes. Figure 4.10 shows the ratio of QR code scans to 1D barcode scans in each country as a PDF. Only a small number of countries have more one-dimensional

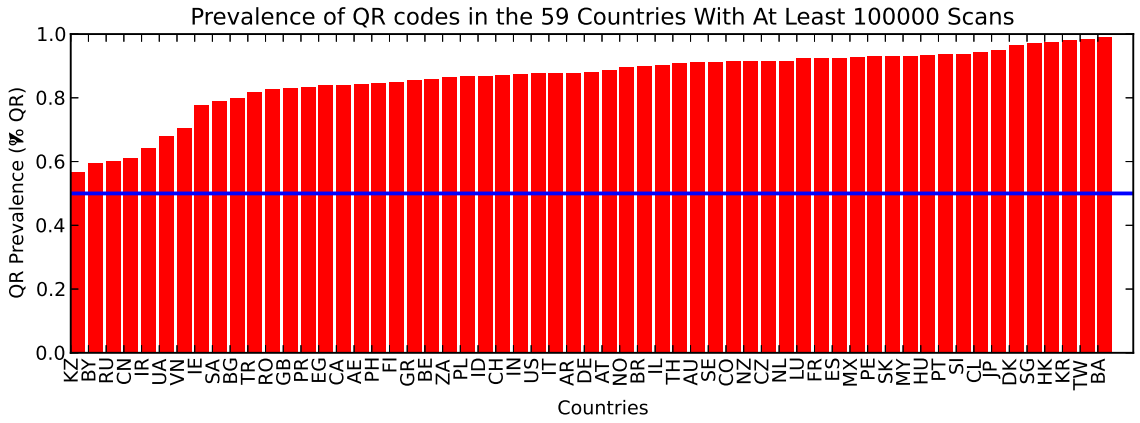



Figure 4.11: Percentage of scans which were of QR codes (as opposed to 1D barcodes) per country for the most scanning countries. The blue line indicates the y-coordinate for 50% QR codes — all countries with at least 100,000 scans are above the line and performed predominantly QR code scans.

barcode scans than QR scans, but there is notable variance even between those countries which are dominated by QR code scans. For example, QR code scans in one country (Bosnia and Herzegovina) make up about 99% of its 631,812 scans, while scans in Russia (4,482,254 total scans) were split 60%/40% between QR codes and one-dimensional QR codes barcodes.

Figure 4.11 shows the same data limited to the countries in which at least 100,000 scans were performed over the studied period. None of these 59 most scanning countries scanned more one-dimensional barcodes than QR codes, suggesting that QR code based use cases dominate among frequent users of the application. We emphasize that our geographical findings may not be representative of the entire QR code usage ecosystem because our results are only from a single application, and the popularity of this application compared to other applications may vary by geographic region.

Thus far, we have analyzed our dataset as a whole, considering distributions of the frequency of scans and types of codes, analyses over time, and geographic variations. In the next sections, we dive more deeply into specific popular and unpopular codes as well as investigate specific use cases of QR codes.

## *4.5 Use Case Analyses*

Having analyzed the overall properties of the dataset, we now turn to analyzing specific use cases of QR codes and barcodes. Our analysis is three-fold. First, we consider the most popular codes (Section 4.5.1). These codes reflect the most common uses of QR codes in our dataset, and hence our analysis of them gives insight into a sizeable and important fraction of the one- and two-dimensional barcode ecosystem. However, our dataset contains many codes that are scanned infrequently (half of all codes in the dataset appear only once). Thus, we also conduct an analysis of infrequently scanned codes, which we define as codes that appear 5 or fewer times in our dataset (Section 4.5.2). Given the diversity of the unpopular codes, our analysis of unpopular codes in Section 4.5.2 is by necessity different than our analysis of popular codes in Section 4.5.1. We cannot, for example, simply pick the 100 least popular codes and analyze each of them (indeed, there are 16,792,603 codes scanned 5 or fewer times). Instead, in Section 4.5.3, we turn to analyzing in more detail specific use cases of QR codes (which span both popular and unpopular codes).

### *4.5.1 Popular Codes*

#### *Contents of Popular Codes*

We first investigate the contents of the 100 most popular (i.e., most frequently scanned) codes. The number of scans of the 100 most popular codes range from 244,660 for the most popular code to 11,925 for the 100th most popular code. The fact that even the most popular codes in the dataset are scanned a relatively modest number of times compared to the total number of scans suggests that users encounter a huge diversity of codes in practice.

We observe that the QR codes real users encounter often contain web links, and that many of the most popular of these links lead to corporate websites. Even among the most popular, however, corporate marketing links do not stand alone, with religious organizations and non-profits like Wikipedia appearing. We also observe a few niche but very popular uses of QR codes, including a code which appears *inside* a video game and a Bitcoin transfer

code.

**Web Codes.** The most frequently scanned codes in the dataset are dominated by links to the web: 95 of the top 100 codes are web links. Most are explicit `http://` links, with a few exceptions: 5 are SSL (`https://`), while 2 are web links that don't specify HTTP or HTTPS explicitly. 40 of the links are to `.com` domains; 8 to `.org`; 3 to `.com.hk` or `.hk`; 4 to `.de`; 5 to `.jp`. At least 15 are shortened links from shortener services and/or QR-code generation services such as `goo.gl`, `bit.ly`, `j.mp`, and `tinyurl.com`, `qrs.ly`, `qr2.it`, and `qrstuff.com`.

The most popular web domains found in the top 100 codes were `jw.org` (Jehovah's Witnesses, 5 of the top 100 codes), `mcd.com`, `mcdonalds.com`, and `happystudio.com` (McDonalds Corporation, including their Happy Studio game, a total of 10/100), and `costco.ms` (3/100). We discuss popular domain in our dataset further below.

**Plain Text Codes.** Three of the most popular 100 codes are plain text, including the following texts: "**", "tpl_not _defined", and a multi-lingual free text message congratulating the person who scans it for having "successfully identified and scanned a QR code! Great job!" which was included in the video game *Guacamelee! Gold Edition* and was scanned 14,558 times by 8098 different devices. The popularity of QR codes displayed in the game speaks to the viability of mobile system applications that use QR codes as an exchange medium between two devices.

The meanings of "**" and "tpl_not_defined" are unclear to the authors of this study. We hypothesize that the latter may come from the QR code at `http://myopenapps.blogspot.com/2014/04/twilight-war-apk-for-android-free.html`, which appears that it intends to be a QR code that links to the associated application. The resulting QR code may have been created in error and published online before testing.

**Other.** The only non-QR code in the top 100 codes (#73) is an EAN-8 code for a bottle of a Coca-Cola product, 54491472. The 90th most popular code, with 13,177 scans, is a `bitcoin:` link specifying a Bitcoin transfer to a wallet belonging to `thepiratebay.sx`. We

discuss the use of QR codes for Bitcoin and other cryptocurrencies in Section 4.5.3 below.

*Popular Domains*

As discussed above, a majority of QR codes contain web URLs. We break these URLs down by domain: Table 4.6 shows the top 50 domains in codes, ordered by the number of times each domain appeared in a code. Table 4.6 also shows the Alexa global rank of each domain for comparison.[3]

Counter to our initial expectation, we find little correspondence between domains that are popular on Alexa and domains that are popular in our dataset. While we do see some of the expected popular sites, including `google.com` and `facebook.com`, we also see a large presence of sites that are unpopular on the web in general. Indeed, 15 of the top 50 domains in our dataset do not even appear in the top 1 million sites on Alexa; an additional 7 domains do not appear in the Alexa top 100,000.

Instead of the conventionally popular websites, we observe increased prevalence of URL shorteners (such as `goo.gl` and `bit.ly`), domains that appear to be specific to QR codes (such as `qrs.ly` and `kaywa.me`), domains related to lotteries (such as `645lotto.net` and `nlotto.co.kr`), and more. These results suggest that the QR code-based web ecosystem is different than the traditional browser based web, with some of the major QR code-based web sites not having a proportionately large presence on the browser-based web.

*Geography of Popular Codes*

The most popular codes in the dataset are quite diverse in terms of geographic diversity. On the one hand, some codes show very high geographic diversity, with the most popular country for one of these popular codes responsible only for about 9% of scans. On the other hand, some codes show almost no geographic diversity. 37 codes are dominated by scans from a single country by a factor of 100: the most popular country for each of these codes

[3]Domain rank data acquired from `http://www.alexa.com` in November 2014. The domain ranks may have differed at the time the codes containing the domains were scanned.

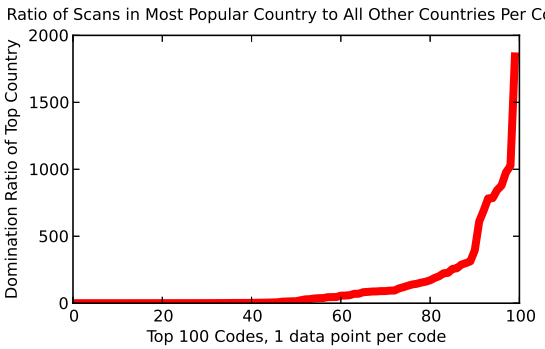



Figure 4.12: The degree of geographic diversity in scans of the top 100 codes. The y-axis represents the degree to which the most scanning country dominates scans from all other countries, represented as a ratio of scans from that country to scans from all other countries.

has scanned the code 100 times more often than all other countries combined. In the most extreme case, the most popular country for a particular code (Japan) has over 1800 times the number of scans (12900 scans) as all other countries combined (12907 combined scans total).

Figure 4.12 shows the ratio of top-country scans to all scans for the 100 most popular codes. For a third of the most popular 100 codes, this ratio is under 1, indicating that no single country was responsible for more than half the scans of these codes. The content of these high-diversity codes are quite diverse, including the link to donate bitcoins to The Pirate Bay, the code from the Guacamelee video game, English language Wikipedia, and a link to a personal blog. 11 of these 33 high-diversity codes lead to pages which are primarily offering the download of mobile apps or which include calls to action to download mobile apps. This suggests that codes with high geographical diversity are more likely to be used in at least two particular cases: more unusual/less corporate uses (religious and non-profits), and to encourage installation of mobile applications.

The most popular code in the dataset (244,660 scans) links to a landing page for a mobile game for iOS and Android. Its scans are dominated by scans from Taiwan (243,450 scans in Taiwan), with a ratio of 200 Taiwanese scans to each scan from any other country (443 scans in the next-closest country, Malaysia). Digging deeper, those scans in Taiwan are

dominated by scans in Taipai, but the code retains significant geographic diversity of scans within Taiwan.

### *4.5.2 Unpopular Codes*

Having considered popular codes, we now turn to unpopular codes. In particular, we consider infrequently scanned codes, which we define to be those that appear 5 or fewer times in our dataset. Note that while the vast majority of codes are infrequently scanned (89% of codes), the scans of those codes account for only 31% of all scans in the dataset, since the frequently scanned codes are often scanned hundreds or thousands of times (see Figure 4.5).

Given the large number of infrequent codes (16,792,603) and their diversity, we cannot simply pick the 100 least frequently scanned codes (as we did for popular codes) and analyze each of them. Rather, we explore now the quantitative differences between frequently scanned, moderately scanned, and infrequently scanned codes.

Specifically, we compare trends of content and usage among codes of differing scanning frequency. For this analysis, we defined "frequently" scanned codes to be those scanned more than 100 times in our dataset, "moderately" scanned codes to be those scanned 6-100 times, and "infrequently" scanned codes to be those scanned 5 or fewer times. This division into bins of 1-5, 6-100 and over 100 scans serves to divide the codes into groups each accounting for approximately one third of all scans in the dataset.

Table 4.7 lists the fraction of codes for each content type (e.g., web, telephone, etc.) that were infrequently, moderately, or frequently scanned. The important comparisons in this table are not across columns — since most codes are scanned infrequently, a given code is most likely to be an infrequent code. Instead, this table allows us to compare across content types in a column: is one content type more or less likely than another content type to be in that frequency category? For example, Android `market:` QR codes are more likely to be scanned moderately than business cards.

### *4.5.3 Specific Use Cases*

We now turn to our analysis of specific uses — and *mis*uses — of QR codes present in our dataset. These use cases, which span both popular and unpopular codes, present a snapshot of different actual uses of QR codes in the wild.

We organize this section into subsections corresponding to five themes: web-related use cases, codes including private data, emerging uses, errors, and malicious uses. A single use case may in fact span multiple themes, but we have organized the section this way to highlight these themes.

#### *Specific Web-Related Uses*

We dive more deeply into two specific web-related use cases: shortened URLs and links to adult websites.

**Shortened URLs.** Many of the web URLs scanned by users were shortened by one of several URL shortening services (e.g., `bit.ly` or `goo.gl`). 984,447 of 11,763,834, or about 8% of URLs seen in the dataset were shortened. To find shortened URLs we checked them against a list of about 40 shortening services that we compiled from sources on the Internet and manually checking URLs which looked like they might be shorteners. A small number of URL shortening services dominated the shorteners seen: 88% of shortened URLs were shortened by Google's URL shortener (`goo.gl`) or by Bitly (`bit.ly`, `bitly.com`). `qr.net`, a service which simultaneously shortens a URL and generates a QR code linking to that shortened URL, appeared as the 7th most popular shortener with 4224 shortened URLs.

We can hypothesize a few purposes for shortening URLs in QR codes. While it may seem odd to shorten URLs in a format which is already only machine readable, we note that URL shorteners are often used for analytics purposes (e.g., to track clicks). Additionally, QR codes with shorter URLs will have less data density, and hence may be easier to scan by some QR code scanners.

**Adult Websites.** We observed a number of scans of URLs leading to websites which serve

adult content. We manually identified 4 domains in the Alexa top 100 whose names clearly indicate that they are adult sites (we may have missed adult sites with less obvious names). We found 7736 scans of 1929 distinct links to these 4 domains.

The human-opaque nature of QR codes makes them a vector for displaying or referencing age- or context-inappropriate material in plain view. We note that for many popular barcode scanners, including the most popular one in the Android Marketplace, the scanner app automatically fetches and displays the title of the specified website. Thus, even if the app doesn't automatically take the user to the site, and even if the site asks visitors for their age (the most popular site in the dataset does not), users will be shown the title of the video or page linked to, which may be inappropriate for them or their context. Further, automatically fetching the website's title will cause the user's device to connect to the site in question without the user's explicit intention.

Our dataset shows that QR codes with links to adult content do exist and are scanned by real users with some regularity. We thus suggest that the designers of QR code scanning applications consider these concerns, perhaps using a list of known adult websites to mask website titles, avoid automatically fetching titles at all, or show additional warnings before redirecting users to such sites.

*Private Data*

We observe a number of use cases that involve encoding private information into a QR code.

**Wifi Setup Codes.** QR codes can be used to encode the information needed to connect a device to a Wifi network. We found 55,809 unique such codes, which were scanned a total of 134,121 times.

For private Wifi networks, these codes contain the plaintext password used to authenticate with the network, exposing that password to anyone with a QR code reader in range of the QR code. As always-on devices with the capability to read QR codes become more prevalent (e.g., Google Glass), we might expect that the threat of (intentional or uninten-

tional) shoulder surfing to read QR codes containing such sensitive information will increase. Another lesson from these results is that designers of applications that emit QR codes should consider the implications of putting private information in the QR codes — an untrustworthy QR code scanner would be able to extract that information. The makers of QR code scanners must also take precautions to protect the privacy of data contained in scanned QR codes.

**Two-Factor Authentication.** We found 79,017 distinct codes using the `otpauth://` scheme, which is used to set up two-factor authenticators (e.g., Google Authenticator). This suggests that two-factor authentication is used by a significant number of people. The URIs included codes to set up authenticators for Microsoft-related accounts (25,753), as well as accounts for Facebook (15,977), Gmail (10,108), Dropbox (2360), Zoho (766), WordPress (741), GitHub (524 codes), DigitalOcean (427), and a large number of others, some of which appeared to be malformed. Most of these codes (78,680) used the standard time-based (TOTP) version of the protocol; the remaining well-formed URIs specify the HMAC-based (HOTP) version of the protocol.

Of the `otpauth://` codes we found, the vast majority (about 99.5%) were only ever scanned by a single device, indicating that only one device was set up to authenticate for that account. Note that unlike Wifi setup codes containing passwords (discussed above), viewing another user's two-factor authentication setup code is not as dangerous, as they do not allow the attacker to compromise the user's account without the primary authentication password. However, they would allow someone to compromise the user's second factor by obtaining the secret used to initiate it.

**PGP Keys.** A QR code containing a public key such as a PGP key is a natural way to convey cryptographic credentials. For example, a person might include their public key as a QR code on their business card to make it easy for acquaintances to import the key and support subsequent communications. Key distribution has long been a topic of interest and study for making cryptographic systems of trust usable and used. Physical artifacts such as

QR codes represent an interesting point in the space of these solutions.

The dataset contains 30 codes representing PGP public keys and 4 codes representing private keys. One of these PGP keys is found embedded in a business card format, while the others stand alone, with the entire QR code representing a full PGP public key block. The public keys were scanned only a total of 56 times, representing a tiny percentage of the scans in the data. The 4 private keys were scanned only 7 times altogether. While we cannot conclude anything in particular from these examples, we note that sharing a private key instead of a public key could be a serious breach of cryptographic security, and that code creator error could result in the inclusion of private keys in scenarios like QR code generation. While these inclusions of private keys could be intentional, the danger of human error in QR code creation is illustrated here. As with Wifi and two-factor setup codes, this finding suggests that QR code creators and consumers should exercise care with private data.

*Emerging and Niche Uses*

**Bitcoin and Other Crypto Currencies.** Our dataset provides us with a glimpse into the use of Bitcoin [131] and other crypto currencies, such as Dogecoin [41] and Litecoin [141]. The `bitcoin:` URI scheme is used for directing a device to make a Bitcoin payment and includes the wallet to pay to and the number of bitcoins to transfer. We found 10,199 codes containing Bitcoin URIs, specifying payments to 8541 distinct Bitcoin wallets. The most popular Bitcoin URI in our data points to `thepiratebay.se`'s Bitcoin address, suggesting that many people made (or considered making) donations to that site. This URI was popular enough to be one of the 100 most scanned codes. Note, however, that the presence of a Bitcoin transaction scan in our dataset does not necessarily mean that the transaction was confirmed by the user and committed to the Bitcoin network.

Bitcoin is significantly more prevalent in our dataset than other cryptocurrencies; we found only 186 Dogecoin URIs and 525 Litecoin URIs.

We also found codes which appear to represent Bitcoin private keys in Wallet Import

Format (WIF) [25]. Keys in WIF are 51 characters long and begin with 5 for private keys. We found 2483 codes in the format of a bitcoin private key, according to the above definition, and verified that 1260 of them are well formatted, i.e., that they can be decoded into private keys which could be used to import the corresponding wallet. We did not import any of them.

**Boarding Passes and Event E-Tickets.** We found electronic tickets, both for transportation as well as for events such as concerts, among the codes scanned in the dataset.

For boarding passes, we found 2416 codes which appear to be in a standardized format for airline boarding passes containing confirmation codes, flight departure and arrival times, airport codes, and names of passengers. These boarding passes were scanned a total of 5396, with 1101 of them scanned only once, 668 scanned twice, and the remaining 647 were scanned more than twice. One boarding pass was scanned 25 times by the same device.

We also found a variety of URLs and eTicket formats for event tickets. We note significant diversity in the formats and strategies used by eTicketing systems. For example, some codes included an `https://` link to a backend ticket processing system which doesn't seem to host public content , while others lead to sites on the public web. These systems seemed not to follow a common standard or format.

There is greater standardization in airline boarding passes than in other eTicketing systems. We attribute this naturally to the need for interoperability between different airlines, security agencies and airports. The contrast between these two systems with similar purposes (admission of a person to a restricted location or event) but differing levels of standardization speaks to the different ways that technologies like QR codes can be used depending on usage context.

**Never Gonna Give You Up.** A popular joke on the Internet is to link unexpectedly to a video of the Rick Astley song "Never Going to Give You Up" (referred to as "rick-rolling") [174]. Because QR codes are not human-readable, they may be an appealing mechanism for delivering this URL to an unsuspecting victim. Indeed, we found 1614

scans of 40 un-shortened codes and 24 shortened codes containing the URL of the video (`https://www.youtube.com/watch?v=dQw4w9WgXcQ`) by devices in 63 countries. The simplest code for a rickroll (with only the URL presented above) was scanned over a thousand times and was likely created by many different people who all chose to create identical codes which were scanned in a diversity of places and at differing times. The most popular country for rickrolling was the United States (with 589 scans), followed by Great Britain and France (with 194 and 154 scans respectively). This suggests that while a few dozen people may have thought to use QR codes in such a joke, each individual creator is unlikely to have spread their code far. While these numbers are small, it suggests that people are using QR codes for innovative, homebrew purposes.

Similar in spirit to rickrolling, we find 993 scans, over 711 different devices and 597 different cities, of the following quote from the movie *Fight Club*: "It could be worse. A female could cut off your Dick while your sleeping and throw it out a moving vehicle." (sic) This QR code, which consisted entirely of text and not a link, also demonstrates that QR codes can be used as a vehicle for communicating directly with humans, rather than first to a mobile device for processing (e.g., rather provide a URI or a PGP key).

**TagMeNot.** Our dataset gives us a unique opportunity to measure the prevalence of emerging uses of QR codes. For example, `TagMeNot.info` is a "pre-emptive, anticipatory, vendor independent, and free opt-out technology for pictures taken in public places" [29]. TagMeNot is an early example of cyber-physical interactions in which aspects of the physical world can be interpreted in digital context. QR codes from TagMeNot indicate that the wearer of the code wishes to opt-out of certain sharing or usage of their likeness or property by the takers of photographs. We found that the code `TagMeNot.info` was scanned only 7 times by 5 different devices in Mexico, Italy, Great Britain, and the United States, suggesting that such opt-out QR codes have not been widely adopted for privacy in the face of ubiquitous cameras.

*Erroneous Uses*

Throughout our analysis of our dataset, we observed a variety of malformed QR codes. In this section, we consider one potentially erroneous use in detail.

**JavaScript.** To our surprise, we observed a number of QR codes containing JavaScript or HTML content. For example, 154 codes, which were scanned a collective 303 times, contained JavaScript code under the URI scheme `javascript:`. Most of these codes are treated merely as plaintext by most QR code readers (i.e., the readers do not attempt to execute the JavaScript, or even offer the opportunity to do so).

We hypothesize that most of these codes suggest a lack of understanding of how to use these code snippets or of the purpose and usage model of QR codes on the part of QR code creators. For example, one of the JavaScript snippets is code for a browser bookmarklet that creates a QR code [92]. Its presence in our dataset suggests that the creator misunderstood how to use this code, which should be pasted into a browser bookmark rather than into a QR code.

*Malicious Uses*

Finally, we consider a number of potential malicious uses of QR codes and investigate their prevalence in our dataset.

**Premium Telephone Numbers.** In the United States and Canada, certain telephone number area codes designate sets of numbers as toll-free or premium numbers [166]. One might hypothesize that QR codes are a vector for tricking people into calling expensive premium numbers. However, we did not find any numbers that we believe to be premium numbers from the US/Canadian system. We did find 667 numbers which appear to be toll free numbers (for example, 1-800 numbers). These numbers were overwhelmingly scanned in the US (91% of scans), suggesting that our guess that these numbers are US toll free numbers is correct.

**Special Phone Numbers.** Phones often respond with special actions, such as displaying

statistics or factory resetting, when special codes are dialed. For example, some Samsung phones display their IMEI number when "*#06#" is typed into the dialer. We find 126 scans of Samsung special codes in our dataset. One such code factory resets certain Samsung devices (*2767*3855#) [126]. Putting such a code into a QR code may be dangerous, since the code must only be displayed in the dialer (i.e., the user must not press the call button) and the device does not ask for confirmation. Indeed, we find 17 scans of this code, suggesting that someone may have attempted to test or actually carry out an attack. Fortunately, none of the scans come from devices of the make and model that treats this code as a factory reset.

**Malicious URLs.** Prior work [168, 176] has suggested that QR codes are a promising vector for distributing malicious URLs. Intuitively, QR codes seem like a natural conduit for phishing attacks or the distribution of malware (e.g., sending users to drive-by-download sites). QR codes are opaque, machine readable pointers, which might make it harder for a user to check that a link is trustworthy. Some QR code apps (including Scan) can be configured to automatically load the URL in a browser without user confirmation. Additionally, the real-world context surrounding a QR code (e.g., its placement on a marketing poster for a trusted brand) might cause the user to trust the code. Our dataset gives us the unique opportunity to study whether users encounter malicious QR code links in the wild.

We investigate this question using Google's Safe Browsing API[4], which provides classifications of a URL as *malware* and/or *phishing*, or *ok*. We randomly sampled URLs in our dataset and queried them against this API (unshortening any URLs shortened by `bit.ly` in the process, using `bit.ly`'s API). Note that the malware/phishing status of a website may change over time (e.g., as a malicious site is taken down or a legitimate site is compromised), but the Safe Browsing API does not provide us with historical data, so our results are necessarily limited.

Of the 1,017,955 unique URLs that we tested, we found 209 URLs flagged as *malware*

[4] `https://developers.google.com/safe-browsing/`

(0.02%) and zero URLs flagged as *phishing*. We were surprised by the latter result, which may suggest that QR codes are presently not a common way to distribute phishing websites. The 209 malicious URLs represent 106 unique domains. Though the fraction of malicious URLs in our dataset is low, the fact that we observe some instances of potentially dangerous URLs suggests that QR code scanning applications should integrate the Safe Browsing API or similar to check URLs before automatically visiting them or allowing the user to visit them.

Note that like QR codes, shortened URLs hide the destination URL and thus might serve as convenient vectors for distributing malware and phishing links. However, similar to our findings, prior work has found that users rarely encounter malicious shortened URLs [113].

**Malicious Android Applications.** Digging deeper into the many web URLs in the dataset, we find that a non-trivial number point to Android applications (i.e., `apk` files). Of QR codes containing web URLs, 49,282 (0.07%) contain such links. Whereas the `market:` URI scheme points to Android applications on the official Google Play app store, `apk` files referenced by web URLs do not come from the official store. Users who have enabled the setting on their Android device allowing application installs from untrusted sources may be prompted to install apps they download through a link.

Since QR codes visually obfuscate links, an attacker may be able to trick an unsuspecting user into installing an Android application in this way. Thus, a natural question to ask is whether any of the `apk` links in our dataset point to malicious Android applications. To investigate this question, we first downloaded each of these `apk` files that was still accessible on the web, and then submitted it to the VirusTotal API for scanning. VirusTotal[5] is a subsidiary of Google that scans files and URLs using multiple antivirus scanners and website engines.

Indeed, we do find instances of known Android malware among the `apk` links in our dataset. We attempted to download the `apk` files from a random sample of 4000 scans

[5] `https://www.virustotal.com/`

containing `apk` links. Of these, 2591 downloads were successful, with the rest failing due to 404 (not found) errors or connection timeouts. We submitted each of these applications to VirusTotal, receiving a result specifying the number of third-party virus scanners checked (generally 40-60) and the number of these that flagged the file. We investigate the VirusTotal report for each application that triggered more than 5 warnings. We find that 26 applications (1.0% of the 2591 apps we downloaded and scanned) are classified as explicit malware (e.g., Trojans). Another 45 applications (1.7%) are classified as Adware, and another 26 (1.0%) are classified as otherwise suspicious. None of these flagged applications appeared twice in the 4000 scans we considered.

We were surprised at the relatively high percentage of `apk` files flagged as malicious or suspicious. Possibly, some users scan QR codes with the intent of downloading applications to help them root their phones (such apps are considered malware by VirusTotal). For example, this video walks a user through the process of rooting their phone and shows a QR code that, when scanned, will download the rooting application: `https://www.youtube.com/watch?v=np18BC6B0OY` (at 1:35). This use case may account for the relatively high number of malicious `apk` files in our dataset.

We spot-checked some of the URLs pointing to malicious `apk` files against the Google Safe Browsing API. The API does not necessarily flag these URLs as malicious. This suggests that it may not be sufficient for a scanning app to check URLs against the Safe Browsing API or similar.

### *4.6 Discussion and Future Work*

We conducted a systematic, in-depth analysis of barcode and QR code usage in the wild. Our results show that barcodes and QR codes, which can enable a physical object to communicate information to a mobile device via a visual channel, are widely used. We analyze that usage in depth. Overall, we believe that our results should be encouraging the researchers and industry practitioners developing new ways of leveraging barcodes and QR codes with mobile devices. We see strong evidence of emerging use cases in our dataset. Our results do,

however, provide a cautionary note: while some users seem to frequently scan codes, other users seem to use their code scanner more as a novelty (with some users only scanning a single code).

We now turn to several lessons from our study, as well as recommendations to the designers of QR and barcode scanning applications.

**Key Lessons.** Stepping back, we summarize the key lessons our analyses reveal about the use cases for QR codes and the frequency with which real users encounter these use cases in practice. These lessons include:

- *QR codes are an effective way to reach some users, but many users are infrequent.* QR codes are still an active technology. However, their use is not universal or uniform: the top users in our dataset performed a stunning number of scans. Half of scans were performed by only 10% of devices, suggesting that this set of users is easily reached by QR codes (e.g., in marketing campaigns). However, many users scan only infrequently: almost one third of devices in our dataset scanned only once. Indeed, we find that despite a steady adoption rate by new devices, the rate of total scans levels out.
- *Web use cases dominate QR codes.* The majority (75%) of scans are of QR codes containing web URLs, suggesting that QR code use is dominated by the use case of quickly connecting users to websites. The popular domains appearing in QR codes do not correspond with domains that are popular on the web in general, with higher popularity of domains specific to QR codes (e.g., `qrs.ly`) and certain use cases (e.g., lottery).
- *Nevertheless, non-web uses are prevalent and varied.* Though non-web codes accounts for only about 25% of all scans (14% of QR code scans), the raw numbers of such scans are still significant. Moreover, the non-web use cases are distributed across a variety of different uses, including Wifi and one-time password setup codes, Bitcoin transactions, and other more niche uses cases. Thus, QR codes appear to be commonly used for a variety of emerging uses.

- *Some codes are intended for broadcast, while others are more limited use.* In comparing frequently and infrequently scanned QR codes, we observe that unique instances of some types of codes (e.g., Android app store URLs) are scanned more frequently than unique instances of other types (e.g., device pairing codes).
- *Scans and codes show no predictable geographic trends.* We observed no consistent geographic distributions of top codes: some are scanned only in one country, others are widely distributed. The cities where we observe the most scans are also geographically diverse. Our caveat about perspective of a single app still applies.
- *QR codes are not commonly used for malicious purposes, but users do encounter some malicious codes in practice.* Though we find that users in our dataset rarely encounter malicious QR codes, our dataset does several examples of malicious QR codes appearing in the wild, including URLs flagged as malicious by Google, links to Android applications containing malware, and a (possibly malicious) factory reset telephone code. Though these cases are rare, users may still encounter them, which informs our recommendations to creators of QR codes and scanning applications below.
- *Some code creators struggle to create correct QR codes.* We observed evidence of malformed QR codes of various types, including QR codes that containing Java-Script intended for a browser bookmarklet. In fact, we hypothesize that one of the top 100 most frequently scanned QR codes was created in error. System designers should not trust code creators to always create correct codes, and researchers should consider code creation as a possible point of failure.

**Recommendations.** Based on our findings, we make the following recommendations to the designers of QR and barcode scanning applications. Scan plans to take these recommendations into account in future version of the application.

*Check for malicious or questionable URLs before automatically opening the link found in a QR code.* Although we find that malicious URLs are uncommon, we nevertheless find some examples of malicious links in our dataset, including links to malicious Android applications,

URLs marked as malware by the Google Safe Browsing API, and telephone codes that can factory reset some devices. We also find a significant number of links to adult sites, which, while not malicious, may not be appropriate for all users and in all contexts. Since users are unable to evaluate the content of a QR code visually, QR code scanning applications should be careful not to automatically load content, including a website's title, for potentially questionable links without first obtaining explicit user consent. We note that checking whether a link is malicious may not always be straightforward — for example, we observed that not all URLs leading to malicious `apk` files are flagged by the Safe Browsing API.

As wearable devices with the capability to quickly and automatically read QR codes, such as Google Glass, become more common, this recommendation will become even more critical. There have already been attacks against Google Glass that take advantage of automatic QR code reading, tricking Glass devices into joining unsafe Wifi networks [62].

*Take steps to protect private information.* Some codes contain private information, such as secrets used to authenticate to Wifi networks, initialize one-time passwords, or access Bitcoin private keys. These codes expose secrets to bystanders, who may be able to intentionally or unintentionally shoulder surf (especially if the bystander has a wearable devices such as Google Glass). Thus, designers of systems that use QR codes should consider the privacy needs of the data in the codes. They also need to consider the set of all potential scanning apps as part of the system's trusted computing base since, if the scanning applications are not trustworthy, the system's privacy properties may not be met. On the scanning side, perhaps the results of a scan of a QR code that contains certain classes of private information (such as private keys) should not be shown to users unless the user (rather than the device) has explicitly initiated the scan; this design would eliminate accidental shoulder surfing.

Users may face significant security risks if the ways in which scanning tools and system designers use QR codes don't match up well with users' expectations of the sensitivity of code data. While system designers should take steps behind the scenes, communicating a better model to users of which actions may be risky and which data may need to be protected

could improve security significantly.

As a result of these findings, Scan plans for future releases to (1) add a user-friendly option to opt-out of data collection and to (2) emphasize their data collection policies directly in the user interface of the app.

**Future work.** We briefly outline directions for future work.

First, an additional investigation of QR code misuse that we may be able to conduct with our dataset is an exploration of QR code based attacks attempting to exploit QR and barcode scanning applications themselves (e.g., via input validation vulnerabilities and other attacks described in related work [36, 91]) or the website linked to in the QR code (e.g., via malicious query parameters). We did not study these exploit attempts in this chapter because, for example, we do not know of a public repository of exploits against different scanning app + operating system configurations. A rigorous analysis might involve running each QR code through different combinations of scanning apps and operating systems, e.g., as was done for web browsers [172, 127].

We would also like to explore how much of the web linked from QR codes is not reachable by crawling the general web. The answer to this question has implications for any application that relies on the reachability of sites via a web crawl. For example, an application like Google Safe Browsing may find webpages to scan based on a crawl and might therefore miss websites linked only in the QR code based web. This gap would in turn pose challenges to QR code scanning applications attempting to test scanned URLs for safety.

This study focused on what we can learn about QR code use from our dataset of real-world scans. It would be valuable to extend our knowledge with a user study that more directly investigated both code creators' and scanners' motivations and experiences with creating and scanning QR codes — e.g., building on [168] to understand how frequently users scan codes in specific physical locations. A user study would also allow us to learn about non-users of QR and barcode scanning applications, a population that is inaccessible in our current dataset by definition.

**Limitations.** Finally, we mention several limitations of our dataset. First, while our dataset is large and and diverse enough that we believe it provides us with general information about the use of QR and barcodes, we are nevertheless limited to this single vantage point of one QR and barcode scanning application. Other applications may be popular among different user groups or in different regions with different behaviors. Similarly, because our dataset ends at a particular date, we expect that devices and codes appearing late in the dataset may be underrepresented; they may become more popular after the end of our dataset. As discussed above, while we use the term "user", we do not strictly have information about users but about devices, which may not map one-to-one onto the set of users.

## *4.7 Conclusion*

One- and two-dimensional barcodes, including QR codes, present a convenient way to link physical objects to digital actions and have been widely adopted in both commercial and academic settings. In this chapter, I have presented work that is, to the best of our knowledge, the first in-depth study of QR and barcodes in the wild, leveraging a unique dataset of 87 million scans from users of Scan, a popular QR code scanning application.

In our analysis, we examined general use patterns of QR and barcodes in the wild, finding that QR codes dominate barcodes and that some users interact frequently with QR codes in the wild whereas other users scan only a single code. We find that a majority of scans contain web URLs, but we also identify a wide range of varied and emerging uses of QR codes, including device pairing and crypto currencies (e.g., Bitcoin). We also identify misuses of QR codes, both by code creators who create malformed codes, as well as potentially intentional malicious behaviors, including links to malware.

Our findings allow us to develop an informed understanding about the types of QR codes being created and how users interact with them in the wild. From these findings, we distill concrete recommendations for QR and barcode scanning applications (e.g., to protect private data and check for malicious URLs). The sheer number of users, scans, and use cases represented in our dataset should be encouraging to researchers and industry practitioners

developing new ways of leveraging QR and barcodes with mobile devices.

Table 4.5: Number of scans in the cities where we observe the most scans.

| *Count* | *Country* | *City* |
|---|---|---|
| 1458830 | TW | Taipei |
| 756964 | HK | Central District |
| 436141 | RU | Moscow |
| 362383 | JP | Tokyo |
| 288478 | US | New York |
| 287529 | MX | Mexico |
| 235983 | CA | Toronto |
| 233458 | US | Chicago |
| 223886 | US | Austin |
| 223373 | US | Houston |
| 220564 | US | Minneapolis |
| 194981 | SG | Singapore |
| 187634 | FR | Paris |
| 180562 | GB | London |
| 179355 | CA | Montreal |
| 171356 | US | San Antonio |
| 170302 | US | Las Vegas |
| 164443 | DE | Berlin |
| 161584 | US | Los Angeles |
| 159301 | US | Virginia Beach |
| 153074 | KR | Seoul |
| 151308 | TW | Nankang |
| 150853 | US | Brooklyn |
| 149544 | US | Charlotte |
| 143165 | DK | Copenhagen |
| 140164 | US | Dallas |
| 137590 | RU | Saint Petersburg |
| 136527 | CH | Full |
| 135315 | US | Arlington |
| 134301 | US | Washington |
| 133922 | DE | Hamburg |
| 130249 | US | San Diego |

Table 4.6: The most popular domains found in URLs in codes scanned. These counts are of distinct codes which included one of these domains — multiple scans of a code are not counted here. Note that the count for `google.com` includes subdomains such as `play.google.com` and `docs.google.com`. The righthand column shows the global Alexa rank for each domain as of November 2014 (values of N/A mean that Alexa does not have data for this domain).

| *Domain* | *Unique Codes* | *Alexa Rank* |
|---|---|---|
| goo.gl | 2732005 | 462 |
| youtube.com | 2474840 | 3 |
| google.com | 1685677 | 1 |
| bit.ly | 1453335 | 4406 |
| facebook.com | 1436226 | 2 |
| apple.com | 1266273 | 35 |
| qrs.ly | 1063665 | 2789080 |
| kaywa.me | 689613 | 3124084 |
| delivr.com | 566530 | 243915 |
| premier-kladionica.com | 527672 | 133052 |
| scn.by | 507066 | N/A |
| 645lotto.net | 374325 | 288448 |
| youtu.be | 369834 | 10515 |
| mcd.com | 347196 | 152098 |
| jw.org | 341542 | 1415 |
| qq.com | 316432 | 10 |
| naver.com | 311335 | 112 |
| bitly.com | 309755 | 388 |
| mta.info | 293387 | 6151 |
| scan.me | 277870 | 140254 |
| nlotto.co.kr | 269148 | 57408 |
| vqr.mx | 266588 | 3349845 |
| tagr.com | 256771 | 13590936 |
| naver.jp | 250224 | 187 |
| towerofsaviors.com | 249296 | 98650 |
| mon-gain.fr | 239418 | 5561369 |
| dropbox.com | 237684 | 85 |
| metrohk.com.hk | 187164 | 81666 |
| bby.us | 181854 | 11907472 |
| tinyurl.com | 165680 | 591 |
| 2d-co.de | 163990 | 17873277 |
| augme.com | 160901 | N/A |
| safeshare.tv | 160412 | 17952 |
| windowsphone.com | 158093 | 1245 |
| phonegap.com | 150780 | 6809 |

Table 4.7: Comparison of QR code contents for three categories of code: frequent, moderate and infrequent codes. We defined infrequent codes to be those with 1-5 scans, moderate codes those with 6-100, and frequent codes those with over 100 scans. The table lists the fraction of codes for each content type that were infrequently, moderately, or frequently scanned. For example, 18.15% of codes leading to Android apps on the market (last row, `market:`) had 6-100 scans (moderate), while a business card is only 1.78% likely to be scanned that many times. The dominance of infrequent codes in general is due to the fact that most codes in the dataset are infrequently scanned — 89% of all codes had 5 or fewer scans (see Figure 4.5).

| Scheme | Total Codes | Frequent % | Moderate % | Infrequent % |
|---|---|---|---|---|
| Business cards (vcard + mecard) | 670359 | 0.01% | 1.78% | 98.20% |
| Dogecoin | 184 | 0.00% | 3.26% | 96.74% |
| BBM (Blackberry Message) | 71985 | 0.01% | 3.45% | 96.54% |
| Litecoin | 525 | 0.19% | 4.38% | 95.43% |
| otpauth (2-factor auth) | 79000 | 0.00% | 4.87% | 95.13% |
| Bitcoin | 10199 | 0.16% | 4.98% | 94.86% |
| Wifi | 55741 | 0.03% | 6.58% | 93.40% |
| iOS apps (`itms-services:` ) | 10179 | 0.14% | 7.51% | 92.36% |
| Barcode | 3647582 | 0.00% | 8.14% | 91.86% |
| Telephone # | 38849 | 0.13% | 8.62% | 91.26% |
| HTTPS | 1507300 | 0.43% | 9.88% | 89.69% |
| SMS | 839 | 0.12% | 10.73% | 89.15% |
| HTTP | 12254403 | 0.53% | 11.13% | 88.33% |
| Android Market (`market:`) | 18544 | 1.15% | 18.15% | 80.70% |
| All (mean over all contents) | 18763779 | 0.4% | 10.2% | 89.4% |

# Chapter 5
# CONCLUSION

In this dissertation, I explored a variety of security and privacy aspects of the web through new measurements and through its relationship to several sister technologies — third-party web tracking, web archives, and QR codes — which surround, expand, reference, and enhance the web. This dissertation used measurement as its primary tool, with the chapters developing a novel measurement technique (Chapter 2), deploying that technique in a new context (Chapter 3), and using a unique dataset to provide new perspectives on the web and these surrounding technologies (Chapter 4) that can help us to understand their current uses, to discover the security and privacy concerns that have arisen and may arise from those uses, and to suggest paths forward.

This dissertation was motivated by the observation that the web has become an incredibly important technology in our society, offering both opportunity and risk as it enhances and exposes both our private and public lives to the Internet. In the preceding chapters, I have presented work which illustrates the need for new developments in technology and policy, and which lays the foundation for those developments. This work — and work that follows it — will help us to ensure that the web and its sister technologies better serve human needs.

# BIBLIOGRAPHY


[1] Disney absorbs Infoseek, July 1999. http://money.cnn.com/1999/07/12/deals/disney/.

[2] Grad School Rankings, Engineering Specialties: Computer, 1999. https://web.archive.org/web/19990427094034/http://www4.usnews.com/usnews/edu/beyond/gradrank/gbengsp5.htm.

[3] Laboratory Corp. of America v. United States, 108 Fed.Cl. 549 (2012), 2012.

[4] People v. Franzen, 210 Cal.App.4th 1193 (2012), 2012.

[5] EX PARTE SERGUEI N. MAMEDRZAEV. 2013 WL 1558372, 2013.

[6] Tharpe v. Lawidjaja, 8 F.Supp.3d 743 (2014), 2014.

[7] The Euroeapn Patent Convention, Article 54: Novelty. https://www.epo.org/law-practice/legal-texts/html/epc/2016/e/ar54.html, 2016. Accessed: 2017-05-17.

[8] Robots.txt meant for search engines dont work well for web archives. https://blog.archive.org/2017/04/17/robots-txt-meant-for-search-engines-dont-work-well-for-web-archives/, 4 2017. Accessed: 2017-05-19.

[9] Summary of s3.amazonaws.com. https://web.archive.org/web/*/http://s3.amazonaws.com/alexa-static/top-1m.csv.zip, 2017. Accessed: 2017-05-05.

[10] Welcome to LexisNexis - Choose Your Path. https://www.lexisnexis.com/en-us/gateway.page, 2017. Accessed: 2017-05-19.

[11] WestLaw.com. westlaw.com, 2017. Accessed: 2017-05-19.

[12] Güneş Acar, Marc Juarez, Nick Nikiforakis, Claudia Diaz, Seda Gürses, Frank Piessens, and Bart Preneel. FPDetective: Dusting the web for fingerprinters. In *20th ACM Conference on Computer and Communications Security*. ACM, 2013.

[13] Gunes Acar, Christian Eubank, Steven Englehardt, Marc Juarez, Arvind Narayanan, and Claudia Diaz. The Web Never Forgets: Persistent Tracking Mechanisms in the Wild. In *Proceedings of the ACM Conference on Computer and Communications Security*, 2014.

[14] Adam Lerner, Anna Kornfeld Simpson, Tadayoshi Kohno, Franziska Roesner. Internet Jones and the Raiders of the Lost Trackers: An Arcahaeological Study of Web Tracking from 1996 to 2016. *25th USENIX Security Symposium*, August 2016.

[15] Gaurav Aggrawal, Elie Bursztein, Collin Jackson, and Dan Boneh. An analysis of private browsing modes in modern browsers. In *Proceedings of the USENIX Security Symposium*, 2010.

[16] Scott G. Ainsworth, Ahmed AlSum, Hany SalahEldeen, Michele C. Weigle, and Michael L. Nelson. How Much of the Web Is Archived? *arxiv.org*, pages 1–10, 2012.

[17] Scott G Ainsworth and Michael L Nelson. Only One Out of Five Archived Web Pages Existed as Presented. *ACM HT'15*, 2004.

[18] Scott G Ainsworth, Michael L Nelson, and Herbert Van de Sompel. Only One Out of Five Archived Web Pages Existed as Presented. In *Proceedings of the 26th ACM Conference on Hypertext & Social Media*, pages 257–266. ACM, 2015.

[19] Istemi Ekin Akkus, Ruichuan Chen, Michaela Hardt, Paul Francis, and Johannes Gehrke. Non-tracking web analytics. In *Proceedings of the ACM Conference on Computer and Communications Security*, 2012.

[20] Mika Ayenson, Dietrich James Wambach, Ashkan Soltani, Nathan Good, and Chris Jay Hoffnagle. Flash Cookies and Privacy II: Now with HTML5 and ETag Respawning. *Social Science Research Network Working Paper Series*, 2011.

[21] Catherine M. Baker, Lauren R. Milne, Jeffrey Scofield, Cynthia L. Bennett, and Richard E. Ladner. Tactile graphics with a voice: Using QR codes to access text in tactile graphics. In *Proceedings of the 16th International ACM SIGACCESS Conference on Computers & Accessibility*, 2014.

[22] A. Barth. HTTP State Management Mechanism, April 2011. `https://tools.ietf.org/html/rfc6265`.

[23] Jason Bau, Jonathan Mayer, Hristo Paskov, and John C. Mitchell. A Promising Direction for Web Tracking Countermeasures. In *Web 2.0 Security and Privacy*, 2013.

[24] A. J. Bernheim Brush, Tammara Combs Turner, Marc A. Smith, and Neeti Gupta. Scanning objects in the wild: Assessing an object triggered information system. In *Proceedings of the 7th International Conference on Ubiquitous Computing (UbiComp)*, 2005.

[25] Bitcoin community. Wallet import format, 2014. `https://en.bitcoin.it/wiki/Wallet_import_format`.

[26] Justin F. Brunelle. 2012-10-10: Zombies in the Archives. `http://ws-dl.blogspot.com/2012/10/2012-10-10-zombies-in-archives.html`.

[27] Justin F. Brunelle. 2012-10-10: Zombies in the Archives. `http://ws-dl.blogspot.com/2012/10/2012-10-10-zombies-in-archives.html`, 2012. Accessed: 2017-05-13.

[28] Justin F Brunelle, Mat Kelly, Hany Salaheldeen, Michele C Weigle, and Michael L Nelson. Not All Mementos Are Created Equal : Measuring The Impact Of Missing Resources Categories and Subject Descriptors. *International Journal on Digital Libraries*, 2015.

[29] Alberto Cammozzo. TagMeNot, 2011. `http://tagmenot.info/`.

[30] Dave Chaffey. The Who, Why and Where of using QR or action codes for marketing. Smart Insights, sep 2012. `http://www.smartinsights.com/mobile-marketing/qr-code-marketing/qr-codes-location-demographic-statistics/`.

[31] Chromium. CookieMonster. `https://www.chromium.org/developers/design-documents/network-stack/cookiemonster`.

[32] Comscore. QR Code Usage Among European Smartphone Owners Doubles Over Past Year, September 2012. `http://www.comscore.com/Insights/Press-`

Releases/2012/9/QR-Code-Usage-Among-European-Smartphone-Owners-Doubles-Over-Past-Year.

[33] Sunny Consolvo, Jaeyeon Jung, Ben Greenstein, Pauline Powledge, Gabriel Maganis, and Daniel Avrahami. The Wi-Fi Privacy Ticker: Improving Awareness and Control of Personal Information Exposure on Wi-Fi. In *12th ACM Conffference on Ubiquitous Computing*, 2010.

[34] Jakub Czyz, Mark Allman, Jing Zhang, Scott Iekel-Johnson, Eric Osterweil, and Michael Bailey. Measuring IPv6 Adoption. *ACM SIGCOMM Computer Communication Review*, 44(4):87–98, 2015.

[35] L. Montulli D. Kristol. HTTP State Management Mechanism, October 2000. https://tools.ietf.org/html/rfc2965.html.

[36] Adrian Dabrowski, Katharina Krombholz, Johanna Ullrich, and Edgar R. Weippl. QR inception: Barcode-in-barcode attacks. In *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices*, 2014.

[37] Linda Deng and Landon P. Cox. LiveCompare: Grocery bargain hunting through participatory sensing. In *Proceedings of the 10th Workshop on Mobile Computing Systems and Applications*, HotMobile '09, 2009.

[38] Tamara Denning, Alan Borning, Batya Friedman, Brian T. Gill, Tadayoshi Kohno, and William H. Maisel. Patients, pacemakers, and implantable defibrillators: Human values and security for wireless implantable medical devices. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2010.

[39] Mohan Dhawan, Christian Kreibich, and Nicholas Weaver. The Priv3 Firefox Extension. http://priv3.icsi.berkeley.edu/.

[40] Mohan Dhawan, Justin Samuel, Renata Teixeira, Christian Kreibich, Mark Allman, Nicholas Weaver, and Vern Paxson. Fathom: a browser-based network measurement platform. In *Proceedings of the ACM Internet Measurement Conference*, 2012.

[41] Dogecoin Project. Dogecoin, 2014. https://dogecoin.org/.

[42] Shawn E. Douglas. Citing from a Digital Archive like the Internet Archive: A Cheat Sheet. http://www.writediteach.com/images/Citing\%20from\%20a\%20Digital\%20Archive\%20like\%20the\%20Internet\%20Archive.pdf. Accessed: 2017-05-08.

[43] Peter Eckersley. How unique is your web browser? In *Proceedings of the International Conference on Privacy Enhancing Technologies*, 2010.

[44] Peter Eckersley. How unique is your web browser? *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6205 LNCS:1–18, 2010.

[45] Editor. Hyphenation exception log. *TUGboat*, 7(3):145, 1986.

[46] Electronic Frontier Foundation. Privacy Badger. https://www.eff.org/privacybadger.

[47] Deborah R Eltgroth. Best Evidence and the Wayback Machine: a Workable Authen-

tication Standard for Archived Internet Evidence. *78 Fordham L. Rev. 181.*, pages 181–215, 2009.

[48] Deborah R Eltgrowth. Best evidence and the wayback machine: toward a workable authentication standard for archived internet evidence. *Fordham L. Rev.*, 78:181, 2009.

[49] Steven Englehardt, Chris Eubank, Peter Zimmerman, Dillon Reisman, and Arvind Narayanan. OpenWPM: An automated platform for web privacy measurement. Technical report, Princeton University, March 2015.

[50] Steven Englehardt, Dillon Reisman, Christian Eubank, Peter Zimmerman, Jonathan Mayer, Arvind Narayanan, and Edward W. Felten. Cookies That Give You Away: The Surveillance Implications of Web Tracking. In *Proceedings of the 24th International World Wide Web Conference*, 2015.

[51] Christian Eubank, Marcela Melara, Diego Perez-Botero, and Arvind Narayanan. Shining the Floodlights on Mobile Web Tracking — A Privacy Survey. In *Proceedings of the IEEE Workshop on Web 2.0 Security and Privacy*, 2013.

[52] Facebook. Facebook's position on "Do Not Track". In *W3C Workshop on Web Tracking and User Privacy*, 2011.

[53] Matthew Fagan. Can you do a wayback on that-the legal community's use of cached web pages in and out of trial. *BUJ Sci. & Tech. L.*, 13:46, 2007.

[54] David Fifield and Serge Egelman. Fingerprinting web users through font metrics. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8975:107–124, 2015.

[55] Python Software Foundation. 21.24. http.cookiejar Cookie handling for HTTP clients, February 2015. `https://docs.python.org/3.4/library/http.cookiejar.html`.

[56] Matthew Fredrikson and Benjamin Livshits. RePriv: Re-Envisioning In-Browser Privacy. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2011.

[57] Karén Gazaryan. Authenticity of archived websites: The need to lower the evidentiary hurdle is imminent. *Rutgers Computer & Tech. LJ*, 39:216, 2013.

[58] Ghostery. Ghostery. `https://www.ghostery.com`.

[59] Avi Goldfarb and Catherine E. Tucker. Privacy Regulation and Online Advertising. *Management Science*, 57(1), January 2011.

[60] Google. Google Glass: Setting up Wi-Fi. `https://support.google.com/glass/answer/2725950?hl=en`.

[61] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Addison-Wesley, 1994.

[62] Andy Greenberg. Google Glass Hacked With QR Code Photobombs. Forbes, July 2013. `http://www.forbes.com/sites/andygreenberg/2013/07/17/google-glass-hacked-with-qr-code-photobombs/`.

[63] Saikat Guha, Bin Cheng, and Paul Francis. Challenges in measuring online advertising systems. In *Proceedings of the ACM Internet Measurement Conference*, 2010.

[64] Saikat Guha, Bin Cheng, and Paul Francis. Privad: Practical Privacy in Online Ad-

vertising. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation*, 2011.

[65] Stephanie Hackett, Bambang Parmanto, and Xiaoming Zeng. Accessibility of Internet websites through time. *ACM SIGACCESS Accessibility and Computing*, page 32, 2003.

[66] Stephanie Hackett, Bambang Parmanto, and Xiaoming Zeng. Accessibility of Internet Websites Through Time. In *Proceedings of the 6th International ACM SIGACCESS Conference on Computers and Accessibility*, Assets '04, pages 32–39, New York, NY, USA, 2004. ACM.

[67] Seungyeop Han, Jaeyeon Jung, and David Wetherall. A Study of Third-Party Tracking by Mobile Apps in the Wild. Technical Report UW-CSE-12-03-01, University of Washington, March 2012.

[68] Seungyeop Han, Vincent Liu, Qifan Pu, Simon Peter, Thomas E. Anderson, Arvind Krishnamurthy, and David Wetherall. Expressive Privacy Control with Pseudonyms. In *SIGCOMM*, 2013.

[69] Aniko Hannak, Piotr Sapieżyński, Arash Molavi Kakhki, Balachander Krishnamurthy, David Lazer, Alan Mislove, and Christo Wilson. Measuring Personalization of Web Search. In *Proceedings of the International World Wide Web Conference*, 2013.

[70] Tian Hao, Ruogu Zhou, and Guoliang Xing. Cobra: Color barcode streaming for smartphone systems. In *10th International Conference on Mobile Systems, Applications, and Services*, 2012.

[71] Eiji Hayashi, Bryan Pendleton, Fatih Ozenc, and Jason Hong. WebTicket: Account management using printable tokens. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2012.

[72] Sunghwan Ihm and Vivek Pai. Towards Understanding Modern Web Traffic. In *Proceedings of the ACM Internet Measurement Conference*, 2011.

[73] Internet Archive. Heritrix Internet Crawler. `https://github.com/internetarchive/heritrix3/releases`.

[74] Internet Archive. The Internet Archive. `https://web.archive.org/web/19970126045828/http://www.archive.org/`.

[75] Internet Archive. Wayback Machine. `https://archive.org/`.

[76] Internet Archive. Internet Archive: Digital Library of Free Books, Movies, Music & Wayback Machine. `https://archive.org/`, 2017. Accessed: 2017-05-12.

[77] Internet Archive. Internet Archive Frequently Asked Questions. `https://archive.org/about/faqs.php\#23`, 2017. Accessed: 2017-05-04.

[78] Internet Archive. Wayback Machine. `https://web.archive.org`, 2017. Accessed: 2017-05-11.

[79] Oded Israeli. 2013 QR Market Summary: Are QR Codes Dead or Alive? Visualead, dec 2013. `http://www.visualead.com/blog/2013-qr-market-summary-are-qr-codes-dead-or-alive/`.

[80] Collin Jackson, Andrew Bortz, Dan Boneh, and John C. Mitchell. Protecting Browser

State From Web Privacy Attacks. In *Proceedings of the International World Wide Web Conference*, 2006.

[81] A. Janc and L. Olejnik. Feasibility and Real-World Implications of Web Browser History Detection. In *Proceedings of the IEEE Workshop on Web 2.0 Security and Privacy*, 2010.

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

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

[84] N. A. John. Sharing and Web 2.0: The emergence of a keyword. *New Media & Society*, 2012.

[85] Shawn M. Jones, Michael L. Nelson, Harihar Shankar, and Herbert Van de Sompel. Bringing Web Time Travel to MediaWiki: An Assessment of the Memento MediaWiki Extension. *CoRR*, abs/1406.3876, 2014.

[86] Samy Kamkar. Evercookie — virtually irrevocable persistent cookies. `http://samy.pl/evercookie/`.

[87] Tai-Wei Kan, Chin-Hung Teng, and Wen-Shou Chou. Applying qr code in augmented reality applications. In *Proceedings of the 8th International Conference on Virtual Reality Continuum and its Applications in Industry*, 2009.

[88] Mat Kelly, Justin F. Brunelle, Michele C. Weigle, and Michael L. Nelson. On the Change in Archivability of Websites Over Time. *CoRR*, abs/1307.8067, 2013.

[89] Mat Kelly, Justin F. Brunelle, Michele C. Weigle, and Michael L. Nelson. On the change in archivability of websites over time. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8092 LNCS:35–47, 2013.

[90] Brian Kennish. Disconnect. `https://disconnect.me/`.

[91] Peter Kieseberg, Manuel Leithner, Martin Mulazzani, Lindsay Munroe, Sebastian Schrittwieser, Mayank Sinha, and Edgar Weippl. QR code security. In *Proceedings of the International Conference on Advances in Mobile Computing and Multimedia*, 2010.

[92] Kirkmc. Create a QR code bookmarklet. Mac OS X Hints, December 2012. `http://hints.macworld.com/article.php?story=20121121054934 96&mode=print`.

[93] Donald E. Knuth. *The TEX book*. Addison-Wesley, 1984.

[94] Donald E. Knuth. *TEX: The Program*. Addison-Wesley, 1986.

[95] Donald E. Knuth. *Computer Modern Typefaces*. Addison-Wesley, 1986.

[96] Donald E. Knuth. *The Metafont book*. Addison-Wesley, 1986.

[97] T. Kohno, A. Broido, and K. Claffy. Remote physical device fingerprinting. In *IEEE Symposium on Security and Privacy*, 2005.

[98] Saranga Komanduri, Richard Shay, Greg Norcie, and Lorrie Faith Cranor. AdChoices? Compliance with Online Behavioral Advertising Notice and Choice Requirements. Technical Report CMU-CyLab-11-00, Carnegie Mellon, March 2011.

[99] Georgios Kontaxis, Michalis Polychronakis, Angelos D. Keromytis, and Evangelos P. Markatos. Privacy-preserving social plugins. In *USENIX Security Symposium*, 2012.

[100] Balachander Krishnamurthy, Konstantin Naryshkin, and Craig Wills. Privacy Leakage vs. Protection Measures: The Growing Disconnect. In *Proceedings of the IEEE Workshop on Web 2.0 Security and Privacy*, 2011.

[101] Balachander Krishnamurthy and Craig Wills. On the leakage of personally identifiable information via online social networks. In *Proceedings of the ACM Workshop on Online Social Networks*, 2009.

[102] Balachander Krishnamurthy and Craig Wills. Privacy Diffusion on the Web: a Longitudinal Perspective. In *Proceedings of the International World Wide Web Conference*, 2009.

[103] Balachander Krishnamurthy and Craig E. Wills. Generating a Privacy Footprint on the Internet. In *Proceedings of the ACM Internet Measurement Conference*, 2006.

[104] D. Kristol and L. Montulli. RFC 2109 - HTTP State Management Mechanism, 1997. `https://tools.ietf.org/html/rfc2109`.

[105] Leslie Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley, 2nd edition, 1994.

[106] Mathias Lécuyer, Guillaume Ducoffe, Francis Lan, Andrei Papancea, Theofilos Petsios, Riley Spahn, Augustin Chaintreau, and Roxana Geambasu. XRay: Enhancing the Web's Transparency with Differential Correlation. In *23rd USENIX Security Symposium*, 2014.

[107] Pedro G. Leon, Blase Ur, Yang Wang, Manya Sleeper, Rebecca Balebako, Richard Shay, Lujo Bauer, Mihai Christodorescu, and Lorrie Faith Cranor. What Matters to Users? Factors that Affect Users' Willingness to Share Information with Online Advertisers. In *Symposium on Usable Privacy and Security*, 2013.

[108] Nektarios Leontiadis, Tyler Moore, and Nicolas Christin. A nearly four-year longitudinal study of search-engine poisoning. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 930–941. ACM, 2014.

[109] Adam Lerner, Alisha Saxena, Kirk Ouimet, Ben Turley, Anthony Vance, Tadayoshi Kohno, and Franziska Roesner. Analyzing the use of quick response codes in the wild. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pages 359–374. ACM, 2015.

[110] Chak Man Li, Pili Hu, and Wing Cheong Lau. Demo: Authpaper - protecting paper-based documents/credentials using authenticated 2d barcodes. In *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, 2014.

[111] Library of Congress. Archived Web Site — Library of Congress. `https://www.loc.gov/websites/`, 2017. Accessed: 2017-05-12.

[112] Arendse Lund. The History of Online Ad Targeting, 2014. http://www.sojern.com/blog/history-online-ad-targeting/.

[113] Federico Maggi, Alessandro Frossi, Stefano Zanero, Gianluca Stringhini, Brett Stone-Gross, Christopher Kruegel, and Giovanni Vigna. Two years of short urls internet measurement: Security threats and countermeasures. In *Proceedings of the 22nd International Conference on World Wide Web*, 2013.

[114] Giorgio Maone. NoScript. http://noscript.net/.

[115] MarketingCharts. Data dive: QR codes, July 2013. http://www.marketingcharts.com/online/data-dive-qr-codes-29525/.

[116] Jonathan Mayer. FourthParty Web Measurement Platform. http://fourthparty.info/.

[117] Jonathan Mayer. Tracking the Trackers: Early Results, July 2011. http://cyberlaw.stanford.edu/node/6694.

[118] Jonathan Mayer. Tracking the Trackers: Self Help Tools, September 2011. http://cyberlaw.stanford.edu/blog/2011/09/tracking-trackers-self-help-tools.

[119] Jonathan Mayer. Safari tracking, January 2012. http://webpolicy.org/2012/02/17/safari-trackers/.

[120] Jonathan Mayer and Arvind Narayanan. Do Not Track. http://donottrack.us/.

[121] Jonathan Mayer, Arvind Narayanan, and Sid Stamm. Do Not Track: A Universal Third-Party Web Tracking Opt Out. Internet-Draft, March 2011. http://tools.ietf.org/html/draft-mayer-do-not-track-00.

[122] Jonathan R. Mayer and John C. Mitchell. Third-Party Web Tracking: Policy and Technology. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2012.

[123] J.M. McCune, A. Perrig, and M.K. Reiter. Seeing-is-believing: using camera phones for human-verifiable authentication. In *IEEE Symposium on Security and Privacy*, 2005.

[124] Aleecia M. McDonald and Lorrie Faith Cranor. Americans' Attitudes about Internet Behavioral Advertising Practices. In *Proceedings of the Workshop on Privacy in the Electronic Society*, 2010.

[125] George R Milne and Mary J Culnan. Using the content of online privacy notices to inform public policy: A longitudinal analysis of the 1998-2001 US Web surveys. *The Information Society*, 18(5):345–359, 2002.

[126] Misterjunky. Samsung S4 Dial pad Phone Codes Discussion. XDA Developers, December 2013. http://forum.xda-developers.com/showthread.php?t=2558791.

[127] Alexander Moshchuk, Tanya Bragin, Steven D. Gribble, and Henry M. Levy. A crawler-based study of spyware on the web. In *Proceedings of the 13th Annual Network and Distributed Systems Security Symposium (NDSS 2006)*, 2006.

[128] Keaton Mowery and Hovav Shacham. Pixel Perfect : Fingerprinting Canvas in HTML5. *Web 2.0 Security & Privacy 20 (W2SP)*, pages 1–12, 2012.

[129] Mozilla. Bug 417800 — Revert to not blocking third-party cookies, 2008. https://bugzilla.mozilla.org/show_bug.cgi?id=417800.

[130] Jamie Murphy, Noor Hazarina Hashim, and Peter OConnor. Take Me Back: Validating the Wayback Machine. *Journal of Computer-Mediated Communication*, 13(1):60–75, 2007.

[131] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System, 2008. `https://bitcoin.org/bitcoin.pdf`.

[132] Ryan Naraine. Windows XP SP2 Turns 'On' Pop-up Blocking, 2004. `http://www.internetnews.com/dev-news/article.php/3327991`.

[133] Ryan Naraine. Google testing login authentication via QR codes. ZDNet, 2012. `http://www.zdnet.com/blog/security/google-testing-login-authentication-via-qr-codes/10105`.

[134] N. Nikiforakis, A. Kapravelos, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna. Cookieless Monster: Exploring the Ecosystem of Web-based Device Fingerprinting. In *Proceedings of the IEEE Symposium on Security and Privacy*, 2013.

[135] Nick Nikiforakis, Luca Invernizzi, Alexandros Kapravelos, Steven Van Acker, Wouter Joosen, Christopher Kruegel, Frank Piessens, Giovanni Vigna, and K U Leuven. You Are What You Include : Large-scale Evaluation of Remote JavaScript Inclusions Categories and Subject Descriptors. *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012.

[136] Nick Nikiforakis, Luca Invernizzi, Alexandros Kapravelos, Steven Van Acker, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. You Are What You Include: Large-scale Evaluation of Remote Javascript Inclusions. In *Proceedings of the ACM Conference on Computer and Communications Security*, 2012.

[137] Nick Nikiforakis, Wouter Joosen, and Benjamin Livshits. Privaricator: Deceiving fingerprinters with little white lies. In *Proceedings of the 24th International Conference on World Wide Web*, pages 820–830. International World Wide Web Conferences Steering Committee, 2015.

[138] Nick Nikiforakis, Alexandros Kapravelos, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. *Proceedings - IEEE Symposium on Security and Privacy*, pages 541–555, 2013.

[139] Mary Emily Ohara. Trump Administration Removes LGBTQ Content From Federal Websites. `https://web.archive.org/web/20170324052626/http://www.nbcnews.com/feature/nbc-out/trump-administration-removes-lgbtq-content-federal-websites-n711416`, 2017. Accessed: 2017-03-27.

[140] Greg Pass, Abdur Chowdhury, and Cayley Torgeson. A Picture of Search. In *Proceedings of the Conference on Scalable Information Systems*, 2006.

[141] Litecoin Project. What is litecoin?, 2014. `https://litecoin.org/`.

[142] James L Quarles III and Richard A Crudo. Using the wayback machine in patent litigation. *Landslide Magazine*, 6(3), Jan/Feb 2014.

[143] Achintya Rao. Using the Internet Archive to cite websites. `https://medium.com/`

@RaoOfPhysics/using-the-internet-archive-to-cite-websites-89bd3f2ce0fd, 2017. Accessed: 2017-05-08.

[144] Research Library of Los Alamos National Laboratory. Time Travel. http://timetravel.mementoweb.org/about/.

[145] Alexey Reznichenko and Paul Francis. Private-by-Design Advertising Meets the Real World. In *Proceedings of the ACM Conference on Computer and Communications Security*, 2014.

[146] Simon Robinson, Jennifer S. Pearson, and Matt Jones. A billion signposts: Repurposing barcodes for indoor navigation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2014.

[147] Franziska Roesner, Tadayoshi Kohno, and David Wetherall. Detecting and Defending Against Third-Party Tracking on the Web. In *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation*, 2012.

[148] Franziska Roesner, David Molnar, Alexander Moshchuk, Tadayoshi Kohno, and Helen J. Wang. World-Driven Access Control for Continuous Sensing Applications. In *ACM Conference on Computer and Communications Security*, 2014.

[149] Franziska Roesner, Christopher Rovillos, Tadayoshi Kohno, and David Wetherall. ShareMeNot: Balancing Privacy and Functionality of Third-Party Social Widgets. *USENIX ;login:*, 37, 2012.

[150] José Rouillard. Contextual qr codes. In *The Third International Multi-Conference on Computing in the Global Information Technology*, 2008.

[151] Myriam Ben Saad and Stéphane Gançarski. Improving the quality of web archives through the importance of changes. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6860 LNCS(PART 1):394–409, 2011.

[152] William Shakespeare. *Hamlet*. F.S. Crofts & Co., Inc., NY, 1946. Act I, Scene 3, Lines 70-72, are apropos.

[153] Mark Simkin, Andreas Bulling, Mario Fritz, and Dominique Schroeder. Ubic: Bridging the gap between digital cryptography and the physical world. In *ESORICS*, 2014.

[154] Tom Simonite. Popular Ad Blocker Also Helps the Ad Industry, June 2013. http://mashable.com/2013/06/17/ad-blocker-helps-ad-industry/.

[155] Natasha Singer. Do-Not-Track Talks Could Be Running Off the Rails. The New York Times, May 2013. http://bits.blogs.nytimes.com/2013/05/03/do-not-track-talks-could-be-running-off-the-rails/.

[156] Ashkan Soltani, Shannon Canty, Quentin Mayo, Lauren Thomas, and Chris Jay Hoofnagle. Flash Cookies and Privacy. *Social Science Research Network Working Paper Series*, August 2009.

[157] Kyle Soska and Nicolas Christin. Automatically detecting vulnerable websites before they turn malicious. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 625–640, 2014.

[158] Kyle Soska and Nicolas Christin. Automatically Detecting Vulnerable Websites Before They Turn Malicious. *23rd USENIX Security Symposium (USENIX Security 14)*, pages 625–640, 2014.

[159] Spivak, M.D., Ph.D. *PCTEX Manual*. Personal TEX, Inc., CA, 1985.

[160] Spivak, M.D., Ph.D. *The Joy of TEX*. American Mathematical Society, RI, 1986.

[161] Steven Englehardt. Do privacy studies help? A Retrospective look at Canvas Fingerprinting. `https://freedom-to-tinker.com/blog/englehardt/retrospective-look-at-canvas-fingerprinting/`.

[162] The National Archives. UK Government Web Archive. `https://www.nationalarchives.gov.uk/webarchive/`, 2017. Accessed: 2017-05-12.

[163] ThreatMetrix. Tech. Overview. `http://threatmetrix.com/technology/technology-overview/`.

[164] Vincent Toubiana, Arvind Narayanan, Dan Boneh, Helen Nissenbaum, and Solon Barocas. Adnostic: Privacy Preserving Targeted Advertising. In *Proceedings of the Network and Distributed System Security Symposium*, 2010.

[165] Blase Ur, Pedro Giovanni Leon, Lorrie Faith Cranor, Richard Shay, and Yang Wang. Smart, useful, scary, creepy: perceptions of online behavioral advertising. In *8th Symposium on Usable Privacy and Security*, 2012.

[166] US Federal Communications Commission. What is a toll-free number and how does it work?, 2014. `https://www.fcc.gov/guides/toll-free-numbers-and-how-they-work`.

[167] Robert Vamosi. Device Fingerprinting Aims To Stop Online Fraud. PCWorld, March 2009. `http://www.pcworld.com/businesscenter/article/161036/`.

[168] T. Vidas, E. Owusu, S. Wang, C. Zeng, L. Cranor, and N. Christin. QRishing: The susceptibility of smartphone users to QR code phishing attacks. In *Proceedings of the 2013 Workshop on Usable Security (USEC)*, 2013.

[169] Thomas Vissers, Nick Nikiforakis, Nataliia Bielova, and Wouter Joosen. Crying wolf? on the price discrimination of online airline tickets. In *HotPETS*, 2014.

[170] Cassie Wagner, Meseret D Gebremichael, Mary K Taylor, and Michael J Soltys. Disappearing act: decay of uniform resource locators in health care management journals. *Journal of the Medical Library Association : JMLA*, 97(2):122–130, 2009.

[171] David Y Wang, Stefan Savage, and Geoffrey M Voelker. Juice: A Longitudinal Study of an SEO Botnet. In *NDSS*, 2013.

[172] Yi-Min Wang, Doug Beck, Xuxian Jiang, Roussi Roussev, Chad Verbowski, Shuo Chen, and Sam King. Automated web patrol with Strider HoneyMonkeys: Finding web sites that exploit browser vulnerabilities. In *Proceedings of the 13th Annual Network and Distributed Systems Security Symposium (NDSS 2006)*, 2006.

[173] Washington Post. From Lycos to Ask Jeeves to Facebook: Tracking the 20 most popular web sites every year since 1996. `https://www.washingtonpost.com/news/the-intersect/wp/2014/12/15/from-lycos-to-ask-jeeves-to-facebook-tracking-the-20-most-popular-web-sites-every-year-since-1996/`.

[174] Wikipedia. Rickrolling. http://en.wikipedia.org/wiki/Rickrolling.

[175] Craig E. Wills and Can Tatar. Understanding what they do with what they know. In *Proceedings of the ACM Workshop on Privacy in the Electronic Society*, 2012.

[176] Huiping Yao and Dongwan Shin. Towards preventing QR code based attacks on android phone using security warnings. In *AsiaCCS*, 2013.

[177] Ting-Fang Yen, Yinglian Xie, Fang Yu, Roger Peng Yu, and Martin Abadi. Host Fingerprinting and Tracking on the Web: Privacy and Security Implications. In *Proceedings of the Network and Distributed System Security Symposium*, 2012.

[178] Harlan Yu. Do Not Track: Not as Simple as it Sounds, August 2010. https://freedom-to-tinker.com/blog/harlanyu/do-not-track-not-simple-it-sounds.

[179] Zack Whittaker. PGP co-founder: Ad companies are the biggest privacy problem today, not governments, 2016. www.zdnet.com/article/pgp-co-founder-the-biggest-privacy-issue-today-are-online-ads/.