Shining the Floodlights on Mobile Web Tracking — A Privacy Survey

Christian Eubank Marcela Melara Diego Perez-Botero Arvind Narayanan
Princeton University
{*cge,melara,diegop,arvindn*}@*cs.princeton.edu*

***Abstract*—We present the first published large-scale study of mobile web tracking.[1] We compare tracking across five physical and emulated mobile devices with one desktop device as a benchmark. Our crawler is based on FourthParty; however, our architecture avoids clearing state which has the benefit of continual observation of (and by) third-parties. We confirm many intuitive predictions and report a few surprises.**

The lists of top third-party domains across different categories devices are substantially similar; we found surprisingly few mobile-specific ad networks. The use of JavaScript by tracking domains increases gradually as we consider more powerful devices. We also analyze cookie longevity by device. Finally, we analyze a curious phenomenon of cookies that are used to store information about the user's browsing history on the client.

Mobile tracking appears to be an under-researched area, and this paper is only a first step. We have made our code and data available at http://webtransparency.org/ for others to build on.

I. Introduction

A. Web privacy measurement

Web measurement has recently emerged as a powerful methodology and an important first step in addressing online privacy violations. There is an inherent information asymmetry in web privacy for two reasons: a lack of adequate disclosure of data collection practices and data use policies by websites, and cognitive limitations of users in understanding these disclosures [10], [1]. Web privacy measurement seeks to correct this asymmetry by detecting, quantifying and sometimes reverse engineering privacy-infringing data collection and use.

There have been numerous notable results in the last year or two, including uncovering Google's practice of bypassing Safari cookie blocking [2], quantification of the top trackers [12], measurement of effectiveness of tools to limit behavioral advertising [3], preliminary evidence of price discrimination [11], a comparison of anti-tracking tools [8], and exposing rampant accidental leakage in social media [6]. These efforts have been highly effective — they have led to better browser privacy tools, given regulators evidence to bring enforcement actions, and put reputational pressure on companies to change their practices.

The key to scalable web privacy measurement is the use of an instrumented browser to interact with websites in an authentic manner, mimicking the behavior of a real user. The browser logs all interactions on each website visited, saving the data for later offline analysis. Fortunately, just such a tool is already available: FourthParty[2], built on top of Firefox [9]. It has already led to some of the results mentioned in the previous paragraph. FourthParty must be used in conjunction with a browser automation framework.

The mobile vacuum. Somewhat surprisingly, this flurry of research has been largely confined to the desktop context, and little is known about web tracking on mobile devices (mobile *app* privacy measurement has attracted some attention: see, for instance, MobileScope[3] and [5], [13]). This is in spite of the fact that mobile devices might be particularly attractive targets for businesses and advertisers because they are attached to individuals, and potentially provide signals like location which can lead to more effective advertising. Mobile devices are gradually catching up to the computing power of desktop computers, so we should expect mobile tracking to become very elaborate in the next few years.

Folk knowledge in the community suggests that this vacuum is because of the relative difficulty of carrying out measurements on smartphones and other devices, given that they are typically locked down in some way and comparatively underpowered. The primary challenge in data collection is the limited programmability of mobile browsers. In addition, researchers must contend with various other limitations including RAM and persistent storage.

B. Our contributions

Our first contribution is the design and implementation of a mobile web privacy measurement tool based on FourthParty. We posit that mobile crawling should be carried out with a client-server architecture, i.e., with the mobile device offloading the storage and computation to a more powerful computer. Within this framework, we discuss several possible designs for browser instrumentation (Section II-B). Our design involved porting FourthParty to Android, and driving the crawl via JavaScript (Section II-D). We are currently workin on merging our fork back into FourthParty.

Second, we use this tool to carry out crawls of the Alexa top 500 websites on one desktop computer and five mobile devices — two tablets, a smartphone, an emulated tablet and

[1]We chose floodlights for our titular metaphor, rather than (say) a spotlight, to emphasize this scalability.

[2]http://fourthparty.info/

[3]http://mobilescope.net/

an emulated smartphone, all running Android (Section II-E). We have made this data (as well as crawling code) available for other researchers to build upon. Our database schema is identical to FourthParty's, which streamlines data analysis.

Third, we use the data collected above to survey the state of mobile web tracking in general and compare it with vanilla (desktop) web tracking. We find that client-side third-party functionality in general is not very complex, as measured by the fact that third-party sites set only about one cookie or make one JavaScript call on average, across all the first-party sites they appear on (Section III-A). We also find that the lists of top destkop and mobile third-party trackers are very similar — we found only two mobile-specific ad networks among the top 100 tracking domains on mobile devices (Section III-B, III-C, III-D). We analyze cookie longevity on various devices in Section III-E.

Finally, we study *growing cookies*, which are third-party cookies that gradually increase in size with repeated visits (to the same third party across multiple first-party sites). While the existence of such cookies is known [7], this is the first time such cookies have been reported and analyzed in the academic literature. A surprising number of third-party sites set at least one growing cookie on desktops, but far fewer do so on mobile. (Section IV). We provide evidence that some of these cookies store the user's browsing history (at the granularity of interest segments, and possibly individual sites visited) and reverse engineer the format in some cases. We argue that automatic reverse engineering of cookie data could be a fruitful direction for future research in web privacy measurement.

II. Design and Implementation

A. Client-Server Architecture

Our data collection architecture (see Figure 1) delegates most of the computation and storage to a supporting server. The device's sole responsibilities are fetching one website at a time and generating a log of its latest interactions (e.g. cookies, JavaScript, embedded HTTP objects). The crawling plugin running on the mobile device sends the interaction log corresponding to the website being visited in the form of SQL statements to the crawling backend running on the supporting server. This way, the amount of state kept in the mobile device's main memory is minimal and the crawl database, which can be several MB in size, is generated on the supporting server's side.

B. Driving the crawl

One architectural decision is how to automate, or "drive," the crawl — FourthParty does not do this; it only intercepts and logs various events and traffic.

WebDriver, recently standardized by the W3C, is a language-neutral API for scripting the browser from a separate process. It was originally defined and implemented by the automation framework Selenium, and has since been

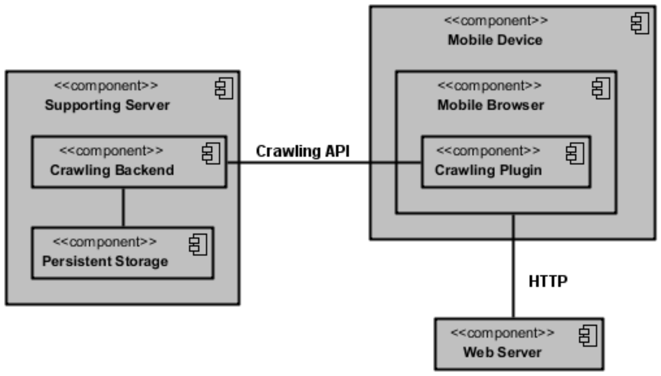


Figure 1: Prototype's Runtime Interactions.

adopted by others like Watir and Marionette. WebDriver is quickly becoming the most popular choice for browser automation and testing.

While WebDriver is very attractive, it does not yet seem to be ready for mobile Firefox (Fennec). Although it has been worked on since 2011,[4], the driver is not yet stable, so we decided against it.

Mozmill is a testing framework similar to Selenium and others mentioned above, but operates more directly on UI elements (e.g., it allows moving the mouse to specified (x,y) coordinates). Mozmill is also an attractive choice in principle, but again does not support Fennec.

Other automation tools we tried were Robocop and Scriptish, in each case running into roadblocks. We have no doubt that one or more of these tools will eventually mature, and allow easy and streamlined automation of mobile Firefox. However, things are still in flux and for now we were left to pursue a simpler approach.

Our approach: JavaScript. We simply use a web page containing a list of URLs and a JavaScript snippet to launch each one in succession. The URLs are launched in a separate tab from the driver, but the same tab is used for each one. A new URL is visited once every 30 seconds.

One benefit of this approach is that it is browser- and platform-agnostic, making comparisons easier. A disadvantage is that it doesn't allow a "deep crawl," say a two-level crawl. The list of websites to visit must be pre-specified.

A second disadvantage is that we cannot isolate visits to different websites within the same crawl. A possible way to fix this is to write an add-on to expose the private browsing API to JavaScript. But once again there were practical difficulties: Firefox's private browsing feature is transitioning from browser-wide to per-window, making the API currently unusable.

On the other hand, non-isolation of websites is also an advantage: we can observe how third parties interact with a user over repeated visits. Indeed, this allowed us to identify

[4]https://wiki.mozilla.org/QA/Goals/Q4

and study "growing cookies" (Section IV).

C. FourthParty SQLite Database Schema

Figure 2 shows the database schema generated on every crawl. The *cookies* table is self-contained and describes all the cookie-related creation, modification and deletion events, along with the host responsible for each event, the value written to the cookie and other interesting metadata. JavaScript execution details are divided into the *javascript* and *javascript_calls* tables, where the former lists the functions triggered by the scripts and the latter stores the arguments given to those functions. On the HTTP side, the *http_requests* table lists each HTTP Request's basic information and is complemented by the *http_request_headers* table, which contains all of the corresponding HTTP headers and their values. A similar arrangement exists for HTTP Responses. Lastly, the *pages* and *content_policy* tables are concerned with the location of the content being sent to the user's browser and the pages responsible for it.

D. Porting

We ported the FourthParty code base to support Android-based mobile devices, such as smartphones and tablets. Our system is implemented in Java and JavaScript, leveraging both the Android SDK and the Mozilla Add-On SDK. Persistent storage is fully compliant with FourthParty's SQLite database schema. Thus, we provide a standardized representation for traditional and mobile crawls, which facilitates data analysis. Our crawling backend is written in Java with a SQLite JDBC library that supports Mac OS, Linux and Windows, so it should be fully multi-platform. It also supports concurrency, so multiple crawls can be recorded simultaneously.

Our code was the end result of four development phases:

1) **Code Refactoring**: The original FourthParty source code had to be refactored to comply with Mozilla Add-On SDK 1.5+ and JavaScript 1.8+. This was done by repeatedly pushing the code to an Android device and analyzing its Exception traces.
2) **Architectural Changes**: The refactored source code was changed to remove its dependencies on local secondary storage and all persistence operations were redirected to a TCP connection.
3) **Support Infrastructure**: The support server and other necessary tools (e.g. crawl scripts generator) were developed.
4) **Testing**: The system was deployed in different devices and various test crawls were conducted to identify and eliminate all remaining bugs.

We are currently working with FourthParty author Jonathan Mayer to contribute support for Android back to the codebase.

E. Web Crawls

For a comprehensive survey of mobile web tracking practices, we conducted six 500-site web crawls through the *Alexa - Top Sites in United States*[5]. Five crawls were conducted on the Android devices described below using the Firefox extension that we developed. We also conducted a crawl using the original FourthParty Firefox extension on a PC; the data collected from the desktop crawl serves as the control for the mobile web tracking practices. We obtained six databases, each offering a rich dataset for later analysis.

All six crawls were conducted between January 21, 2013 and February 10, 2013 on the same set of 500 URLs obtained from the *Alexa - Top Sites*.

F. Devices

We ran our crawls using one PC, one smartphone and two tablets, as well as an emulated smartphone and an emulated tablet:

- Desktop (Ubuntu 12.04, Firefox 11.0)
- Asus Transformer Pad TF300T (10.1-inch Tablet)
- Samsung Galaxy Tab 2 (7.0-inch Tablet)
- HTC Evo 4G (4.8-inch Smartphone)
- Emulated Nexus 7 (7.0-inch Tablet)
- Emulated Nexus S (4.0-inch Smartphone)

The two emulated devices were created with the Android Virtual Device (AVD) tool that comes with the Android SDK. We modified the emulator's settings in order to create an Android smartphone and tablet that were as close as possible to their physical equivalents. Four non-generic system images are packed with the Android 4.2 API SDK tools: Nexus 7 (tablet), Galaxy Nexus (phone), Nexus S (phone) and Nexus One (phone). Therefore, we believe that they provide the most accurate runtime environments when trying to impersonate physical devices. Out of all the phone images, the Nexus S shared the most similarities with the HTC Evo 4G in terms of their technical specifications, so it was deemed more accurate than the other two options.

Why use emulated devices rather than simply spoofing the User-Agent on a physical device? While websites do often customize the interface simply based on the User-Agent, it is considered better practice to utilize specific properties such as screen resolution instead. Using emulated devices is therefore more authentic than User-Agent spoofing. That said, it would be interesting to measure how different the two actually are; we did not do it since it is outside the scope of our privacy study.

As we have covered a broad range of popular Android devices currently in the field, we believe our collected data is a good representation of web tracking practices today.

[5]Our decision to focus on the Top US Sites instead of the Top Global Sites came after a series of crashes were caused by websites containing non-western character sets

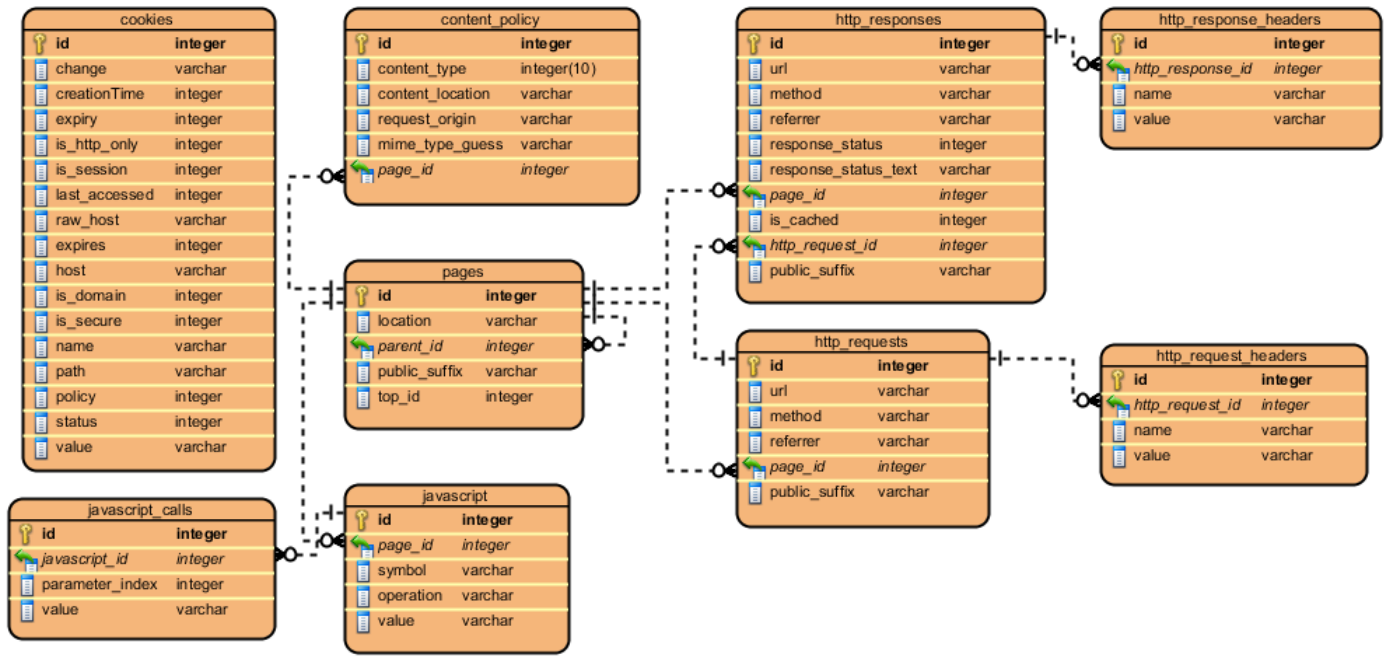


Figure 2: FourthParty SQLite database schema

III. Data Analysis: Desktop vs. Mobile

A. Overview

To gain a general perspective of our collected data and characterize it based on the presence of first-party versus third-party domains, we computed various ratios involving first- and third-party domains, cookies and JavaScript function calls.

Complexity of functionality. We first calculated the average number of unique cookies added per first-party domain and per third-party domain. We also separately compute the latter average over only the top 500 third-party domains (there are about 1,000 third-party domains overall in our dataset). We performed the analogous calculations for the unique JavaScript function calls. We consider a unique cookie to be a unique $(cookieName, domain)$ pair, while a unique JavaScript function call is a unique $(functionName, domain)$ pair.

First and third parties add more cookies on average on the desktop than on the mobile devices, which is probably due to the limited local storage on mobile devices. To our surprise, first-party domains on average store significantly more cookies and make significantly more JavaScript calls than third-party domains (in each case, by about an order of magnitude) on every device studied (Tables I and II). This reveals that the functionality of third parties is rather simple, especially on mobile platforms — each third party adds about one unique cookie. With respect to unique JavaScript calls, third-party functionality on the desktop is more complex, but the complexity decreases dramatically on the mobile devices we studied.

The average top 500 third-party domain make more JavaScript calls and adds more cookies than the average third-party domain overall, which suggests that more popular third-party domains host more complex functionality

Device	Avg Unique Cookies		
	1P	All 3P	Top 500 3P
Desktop	10.5	1.4	2.4
Tab. (Asus)	6.8	1.1	1.4
Tab. (Galaxy)	8.6	1.4	1.9
Phone	7.5	1.1	1.2
Em. Tab.	6.3	0.9	1.0
Em. Phone	7.2	1.0	1.1

Table I: Average number of unique cookies added per domain.

Device	Avg Unique JS Calls		
	1P	All 3P	Top 500 3P
Desktop	24.4	5.8	9.3
Tab. (Asus)	4.5	1.3	1.9
Tab. (Galaxy)	10.2	2.4	4.0
Phone	4.7	1.3	1.8
Em. Tab.	4.0	1.1	1.5
Em. Phone	4.5	1.3	1.7

Table II: Average number of unique JavaScript function calls made per domain.

Type of functionality. Another way to get a general overview of trackers is to categorize their functionality. This is hard to automate; Roesner et al. provide a methodology for doing so [12] but it is not clear if this can be computed from FourthParty logs.

However, we can get an intuition for tracker types by looking at the fractions of third-party domains which add cookies or make JavaScript calls (see Table III). Note that the majority of domains (51.0% for desktop, 56.4% on average for mobile) do neither, suggesting that they merely serve content.

An interesting implication is that the majority of third-party domains are not thought of as trackers in the usual sense (since they don't employ tracking cookies or JavaScript), but they nevertheless have access to protocol logs (IP, HTTP, etc.) which frequently uniquely identify the user. Should policies or regulations on web tracking only focus on explicit trackers, or all third parties? We offer no opinion, but merely note that there is a perhaps surprisingly large fraction of domains to which this question is relevant.

Another surprise is that the numbers are roughly the same between desktop and mobile, even though the number of JavaScript calls per domain is much higher than on desktop than on mobile, as we saw earlier.

	Third-Party Domains (in %)			
Device	**Cookie**	**JS**	**Either**	**Both**
Desktop	33.2	21.8	49.0	6.0
Tab. (Asus)	29.2	16.9	44.3	1.8
Tab. (Galaxy)	32.6	15.7	45.5	2.8
Phone	30.7	17.3	45.0	3.0
Em. Tab.	25.9	17.2	40.5	2.7
Em. Phone	29.5	15.8	42.6	2.7

Table III: Proportion of third-party domains which add cookies, make JS function calls, do either or do both.

B. *Top Third-Party Domains*

We identified the top third-party domains for all devices, and then compared their rankings. For this purpose, we chose to do a comparison among the top 20 third-party domains for all devices as we expected to see a few clear contenders for all devices and some interesting variations at the same time.

We found that `google-analytics.com` is the number one third-party domain for all six studied devices. The other two highest-ranking third-party domains for all six devices are `doubleclick.net` and `scorecardresearch.com`.

There are a total of 25 distinct third-party domains in the top 20 across all six devices. Using the desktop third-party domains as the control set, we found that all six devices have 15 of the desktop top 20 third-party domains in common (see Table IV). In terms of domains unique to one device class or another, we found that `invitemedia.com` is unique to the desktop top 20 third-party domains with a ranking of 17, while `googletagservices.com` is unique to the mobile top 20 third-party domains with an average ranking of 13.8.

In summary, the top trackers on mobile and desktop devices were much more similar than we expected.

Domain	**Average Ranking**
`google-analytics.com`	1
`scorecardresearch.com`	2.2
`doubleclick.net`	3.7
`googleapis.com`	4.2
`g.doubleclick.net`	4.3
`quantserve.com`	6
`facebook.net`	6.7
`ak.facebook.com`	8.5
`2mdn.net`	9.8
`cloudfront.net`	9.8
`imrworldwide.com`	12.8
`googlesyndication.com`	13.2
`yieldmanager.com`	14
`atdmt.com`	15.2
`revsci.net`	18.2

Table IV: Top-20 third-party domains common to all six studied devices

C. *Mobile-Only Third Parties*

As noted above, desktop and mobile devices share most of their top third-party trackers. However, given the ubiquity and variety of mobile devices, intuition suggests that a new set of tracking companies/domains would emerge, exclusive to the mobile advertising and analytics space. To test this hypothesis, we sought to identify third parties that are only present on mobile devices through a simple metric. We call a third party mobile-only if it appears in the list of top 100 third party sites for any of the mobile devices but not in the top 500 for desktop. The rationale is that if a tracker is (say) the 100th by rank on a mobile device and 200th on desktop, the difference could be explained away by statistical noise leading to differences in ranking.

Contrary to our intuition, we found that each physical and emulated mobile device contained only a handful of top 100 third party sites that did not also appear in the desktop top 500. These sites were typically the mobile versions of third party sites present on the desktop. The two exceptions were `admarvel.com` and `mocean.mobi`, advertising networks centered around mobile devices.

The dearth of third parties that exclusively focus on mobile devices is surprising. Perhaps already-established third parties have transitioned to mobile tracking or new third parties have simply not yet entered this relatively new market. Regardless of the reason, our metric for detecting mobile-only third parties can be utilized for keeping track of the growth of this market in the future.

D. Similarity Measures

As part of a more fined-grained examination of the differences between desktop and mobile tracking, we compare specific behaviors between devices. Although we have already found that very few third parties have an exclusively mobile presence, individual third parties may still be behaving differently across platforms in terms of their placement of individual cookies or JavaScript calls they make. We measure this using the following two similarity measures, applied to the observations of the *same site* on two different devices.

1) Cookie similarity: The cosine similarity between the set of (first or third party) cookie names for a given domain on two devices.[6] For example, doubleclick.net sets the cookies `_drt_`, `id`, `nmfirstparty`, `rsi_segs`, and `test_cookie` on desktop, but only three of these mobile, making the cosine similarity 0.77.
2) JavaScript similarity: The cosine similarity between the set of JavaScript function calls (names) invoked by the domain on the two devices. We only count `window.LocalStorage.*` and `window.SessionStorage.*` calls because these are more indicative of tracking.

For each pair of devices, after computing a sequence of per-site similarity values, we average them together to compute a single similarity score for a pair of devices. The results for cookie similarity are shown in Table V. The results for JavaScript similarity are omitted because they were essentially identical.

Device	Desktop	Asus	Galaxy	Phone	E. Tab.	E. Phone
Desktop	1.00	0.77	0.82	0.76	0.75	0.75
Asus		1.00	0.87	0.83	0.84	0.83
Galaxy			1.00	0.88	0.88	0.87
Phone				1.00	0.86	0.95
E. Tab.					1.00	0.86
E. Phone						1.00

Table V: Third party cookie similarity across devices

On the whole, the devices are roughly equally similar to each other with one exception: the physical and emulated phones are much more similar to each other than the other devices. Observe that this trend does not hold for the tablets. In particular, the Galaxy tablet has essentially equal similarity measures with the Asus tablet, phone and two emulated devices.

E. Cookie Longevity

While the content stored in cookies is obviously relevant for gauging third party tracking habits, the expiry length for cookies could also reveal third party intent. For instance, third parties have an incentive to create cookies that have more time to gather information about a given user. We discuss this notion further in a section about cookies that grow over time.

Device	**Party**	
	First Party	**Third Party**
Desktop	0.96	0.87
Tablet (Asus)	0.59	0.73
Tablet (Galaxy)	0.93	0.89
Phone	0.93	0.89
Emulated Tablet	0.88	0.65
Emulated Phone	0.91	0.88

Table VI: Proportion of sites that placed long-lived cookies

We call a cookie *long-lived* if it is a persistent cookie (not a session cookie) and its expiry time is over a month from creation time. The proportions of sites that placed at least one long-lived cookie is contained in Table VI. Although the raw number of cookies added during the crawls varied across devices, four of the six devices were such that the proportion of sites that added long-lived cookies were essentially identical.

At first glance, the fact that a greater proportion of first party sites placed long-lived cookies when compared to third parties appears surprising. However, this can be explained by first parties adding long-term cookies to save a user's state on the website (e.g. credentials).

As a more fine-grained metric of cookie longevity, we calculated the mean of logarithms expiry times averaged across sites for each device.[7] For the purposes of analysis, we assigned an expiry length of 30 minutes for all session cookies and an expiry length of one month for all cookies with an expiry length above this threshold. These results are contained in Table VII.

In accordance with intuition, cookies placed by third party sites have longer expiry lengths than those placed by first party sites. As previously mentioned, while first party sites might be more likely to place at least one long-lived cookie, this log mean analysis demonstrates that, on the whole, third parties place cookies have a greater degree of longevity.

Next, note that the emulated and physical phones had greater expiry lengths when compared to the desktop. The results for the two physical and emulated tablets are a bit more dispersed. A plausible reason for the increased longevity of first-party cookies on phones is that it is annoying to login on phones when a login expires. The reason for increased longevity of third-party cookies on phones is not clear.

IV. Growing Cookies

In the vast majority of cases, a cookie's value field consists of a string or integer that remained largely static

[6]The cosine similarity between two sets X and Y is $\frac{|X \cap Y|}{\sqrt{|X||Y|}}$.

[7]It is not meaingful to average the expiry times directly since this can be thrown off by outliers.

Device	Party	
	First Party	Third Party
Desktop	5.19	5.53
Tablet (Asus)	4.55	5.35
Tablet (Galaxy)	5.37	5.71
Phone	5.31	5.75
Emulated Tablet	5.29	4.99
Emulated Phone	5.27	5.73

Table VII: Means of base-10 logs of cookie expiry times. For example, a mean of 5.5 corresponds to a time of $10^{5.5}$ seconds, or 3.7 days. Also note that an additive difference of 0.5 corresponds in this table to a roughly threefold difference in expiry time.

throughout the duration of a particular crawl. However in certain cases, we observed that the values for third party cookies consistently increased in size across successive changes. While some of these growing values are Base64 encoded strings whose decoded binary values do not present an obvious pattern over time, other values grow in a very specific manner.

In particular, these values appear to be lists related to user's browsing history. Figure 3 presents a brief analysis of a representative example. Each node of the list is delineated by separator tokens and is comprised of two values separated by another type of token. It appears that the first value contained in each node is either an ID corresponding to each visited site, or an ID for a category of the site (essentially, an interest segment in advertising terminology).

We hypothesize that the second value in each node is a site-specific ID for the individual user. While the latter value does not have enough entropy to be a unique user ID (i.e., unique among all the users that the third party has encountered), it has enough bits to be unique when combined with the approximate creation time of the cookie.

We base our claims on two key observations. First, the growth of these cookies is characterized by the addition of new nodes somewhere in the string each time a site is visited. Second, after repeatedly running new crawls through sites known to place one type of the aforementioned cookies, the nodes corresponding to a particular site were added in the order in which we visited them.

The first node value, which is almost certainly a site or segment ID, remained constant across the different crawls. For some growing cookies these IDs had a few duplicates (different first-parties with the same ID), which rules out site-specific IDs and suggests segment IDs. For other growing cookies we did not observe duplicates, although this does not rule out the possibility of duplicates if we observed more first parties beyond the top 500. The second value in each node, which we believe to be the user ID, changed across crawls in accordance with our hypothesis.

Nevertheless, new nodes were not added to history lists in a particular order, and sometimes the entire list could be shuffled between changes. The most likely explanation is that the histories are stored in a JavaScript (associative) array before being serialized into a cookie. Associative arrays do not preserve the order of insertion.

Observe that these history-storing cookies expose a privacy vulnerability. Suppose an attacker uses cross-site scripting to read the contents of one of these growing cookies. Then, he could a lookup table for the cookie's site ID's to recover (an approximation to) the victim's browsing history. An even weaker adversary could simply listen in on the communication between the user and website to intercept the contents of these cookies.

Having established the existence of growing cookies, for each device we then examined the proportion of top 500 third party sites that place growing cookies during the crawls. The results are contained in Table VIII.

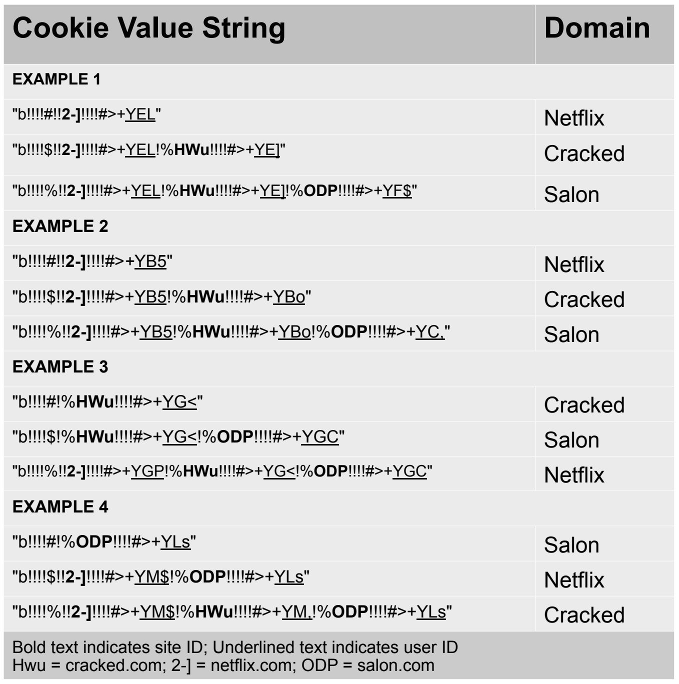
Formally, we consider a cookie to be a growing cookie if it had been changed at least five times and if it satisfies a certain growth metric. We consider the standard growth metric, which we denote `X3`, to be an indicator of whether over the course of the crawl, the cookie's value tripled in length and had a final length of at least 25 characters. We imposed the limit of 25 characters so as to not count the common case of cookies that are added with either no value or a single-character value that is immediately changed to a short but static string. Clearly these types of cookies more than triple in size but do not grow beyond this initialization. As a stricter requirement for growth, which we denote `Strict`, we consider the subset of cookies that pass the `X3` requirements but have at least 75% of their changes increasing their value's lengths and no more than 10% of changes decreasing these lengths.

Device	Growing Metric	
	X3	X3 w/ Strict
Desktop	8.0%	3.6%
Tablet (Asus)	3.2%	0.8%
Tablet (Galaxy)	4.4%	2.6%
Phone	1.0%	0.4%
Emulated Tablet	0.4%	0.2%
Emulated Phone	0.8%	0.6%

Table VIII: Percentage of top 500 third parties using growing cookies

We find that regardless of the metric, a much larger proportion of top third party sites place growing cookies on the desktop when compared to the mobile devices, with perhaps the slight exception of the Galaxy tablet. Both the emulated phone and physical phone have extremely low proportions for growing cookies.

Overall, the desktop has a greater proportion of growing cookies among top third party sites for two possible reasons. First, the relative immaturity of mobile tracking when compared to desktop tracking might mean that trackers have not yet fully ported their growing cookie infrastructure to mobile



Cookie Value String	Domain
EXAMPLE 1	
"b!!!!#!!**2-]**!!!!#>+YEL"	Netflix
"b!!!!$!!**2-]**!!!!#>+YEL!%**HWu**!!!!#>+YE]"	Cracked
"b!!!!%!!**2-]**!!!!#>+YEL!%**HWu**!!!!#>+YE]!%**ODP**!!!!#>+YF$"	Salon
EXAMPLE 2	
"b!!!!#!!**2-]**!!!!#>+YB5"	Netflix
"b!!!!$!!**2-]**!!!!#>+YB5!%**HWu**!!!!#>+YBo"	Cracked
"b!!!!%!!**2-]**!!!!#>+YB5!%**HWu**!!!!#>+YBo!%**ODP**!!!!#>+YC,"	Salon
EXAMPLE 3	
"b!!!!#!%**HWu**!!!!#>+YG<"	Cracked
"b!!!!$!%**HWu**!!!!#>+YG<!%**ODP**!!!!#>+YGC"	Salon
"b!!!!%!!**2-]**!!!!#>+YGP!%**HWu**!!!!#>+YG<!%**ODP**!!!!#>+YGC"	Netflix
EXAMPLE 4	
"b!!!!#!%**ODP**!!!!#>+YLs"	Salon
"b!!!!$!!**2-]**!!!!#>+YM$!%**ODP**!!!!#>+YLs"	Netflix
"b!!!!%!!**2-]**!!!!#>+YM$!%**HWu**!!!!#>+YM,!%**ODP**!!!!#>+YLs"	Cracked
Bold text indicates site ID; Underlined text indicates user ID Hwu = cracked.com; 2-] = netflix.com; ODP = salon.com	

Figure 3: This figure contains the cookie value growths for `yieldmanager.com`'s history-storing `bh` cookies as recorded after visiting three sites known to place these cookies in different orders across four different crawls. Individual website nodes appear to be separated by the token `%!` while the site/segment ID and user ID are separated by `!!!!#>+`. Observe that the nodes containing site/segment IDs are added in the order in which the corresponding sites are visited. The associated strings that we believe to represent user IDs for each site remain constant within each crawl but vary across crawls. This is especially apparent when comparing Examples 1 and 2. Note that the nodes for each website appear in the same relative ordering (specifically `netflix.com` precedes `cracked.com` precedes `salon.com`). This particular ordering suggests that the third party represents the history nodes in a JavaScript associative array, accounting for the fact that the nodes consistently appear in a specific sorted order that is independent of the order in which they were visited.

devices. Second, with certain sites placing multiple growing cookies on a single device and with each of these cookies reaching sizes of potentially over 4 KB, third party sites may instead be leveraging history recording methods that are less memory-intensive for mobile clients.

Automated reverse engineering? Our manual analysis above hints at several underlying principles that we utilized:

1) Differencing: observing changes in client-side state. This could be between different websites as the same user, the same website as a different user, or simply a different order of visits.
2) Reverse engineering of data structures. This involves identifying encodings, seprators, etc. There is a rich literature on reverse engineering of file formats; our example is rather simple in comparison.
3) Studying JavaScript–cookie interaction. We used our knowledge of how arrays are implemented in JavaScript to arrive at a hypothesis just by looking at cookies, but a more sophisticated approach would be to build an analysis engine that would execute JavaScript code and examine how cookie data gets represented in memory.
4) Entropy analysis. Determining whether a given cookie has enough entropy (based on maximum length as well as how it actually changes) to store a certain type of information (site ID, segment ID, user ID, etc.)

We can imagine these principles being the foundation of an automated cookie reverse-engineering mechanism (and beyond cookies, all other client-side functionality). Clearly it would have important implications for web privacy by shining the light on how personal data is collected and used with only minimal human analysis, but more generally it would bring transparency to the web and inform major policy debates ranging from price discrimination to the filter bubble.

V. Future Work

More platforms and other enhancements. Our data collection framework currently exists only as an add-on for Firefox Mobile on Android devices. To gain a better perspective of mobile web tracking on other popular devices, it needs be modified to operate with other platforms such as iOS, BlackBerry, and Windows Mobile.

Other limitations of the current architecture that could be addressed in future versions are: a direct comparison between emulated and corresponding physical devices (we were not able to obtain virtual images corresponding to any of the physical devices we had access to); repeating our measurements with state cleared between visits of different websites; fixing the crashes on non-Western character setsand including those websites in the crawls.

Further data analysis. The main focus of our work was to conduct a preliminary survey of the mobile web tracking realm. There is much more than can be analyzed with respect to the actual values of cookies, JavaScript calls and HTTP request- and response-headers. It may be helpful for future researchers to categorize the top 500 third-party domains to obtain a fine-grained classification of the top trackers into advertising domains, content distribution networks and analytics companies, etc.

Related projects. As discussed in Section IV, one promising avenue for future work is automated reverse engineering of cookie data and client-side functionality in general. Another direction is the use of machine learning for automatic tracking defenses. Bau et al. have communicated early results along these lines (in fact, also submitted to this workshop) [4].

VI. Conclusion

At a time of unprecendented technical, press and policy attention to web privacy, the mobile web is quickly maturing. Given the success of web privacy measurement in general, the importance mobile web privacy measurement is unquestionable.

In this work we've taken the first step in making large-scale mobile web privacy measurement a reality. Our work involved porting browser instrumentation tools to Firefox on Android, data collection from various devices, various analyses comparing desktop vs. mobile tracking, and a study of growing cookies. We've also laid out several directions for future work.

This study is part of a much larger, nascent effort at Princeton on bringing transparency to the web by detecting, quantifying and reverse engineering algorithmic practices involving personal data online. Our goal is to study and influence both privacy and fairness (e.g., price discrimination, political targeting, "filter bubble," and many other such practices). The interested reader is welcome to contact the authors.

Acknowledgments

The authors would like to thank Jonathan Mayer for useful technical discussions and comments on a draft of this paper.

References

[1] Alessandro Acquisti. Privacy in electronic commerce and the economics of immediate gratification. In *Proceedings of the 5th ACM conference on Electronic commerce*, pages 21–29. ACM, 2004.

[2] Julia Angwin and Jennifer Valentino-Devries. Google's iPhone tracking. Wall Street Journal. http://online.wsj.com/article/SB10001424052970204880404577225380456599176.html.

[3] Rebecca Balebako, Pedro Leon, Richard Shay, Blase Ur, Yang Wang, and L Cranor. Measuring the effectiveness of privacy tools for limiting behavioral advertising. In *Web 2.0 Security and Privacy Workshop*, 2012.

[4] Jason Bau, Jonathan Mayer, Hristo Paskov, and John. Mitchell. A promising direction for web tracking countermeasures. Manuscript.

[5] Michael C Grace, Wu Zhou, Xuxian Jiang, and Ahmad-Reza Sadeghi. Unsafe exposure analysis of mobile in-app advertisements. In *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*, pages 101–112. ACM, 2012.

[6] Balachander Krishnamurthy and Craig E Wills. On the leakage of personally identifiable information via online social networks. In *Proceedings of the 2nd ACM workshop on Online social networks*, pages 7–12. ACM, 2009.

[7] Jonathan R. Mayer. Tracking the trackers: Early results. https://cyberlaw.stanford.edu/blog/2011/07/tracking-trackers-early-results.

[8] Jonathan R. Mayer. Tracking the trackers: Self-help tools. https://cyberlaw.stanford.edu/blog/2011/09/tracking-trackers-self-help-tools.

[9] Jonathan R. Mayer and John C. Mitchell. Third-party web tracking: Policy and technology. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, SP '12, pages 413–427, Washington, DC, USA, 2012. IEEE Computer Society.

[10] Aleecia M McDonald and Lorrie Faith Cranor. Cost of reading privacy policies, the. *ISJLP*, 4:543, 2008.

[11] Jakub Mikians, László Gyarmati, Vijay Erramilli, and Nikolaos Laoutaris. Detecting price and search discrimination on the internet. In *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, pages 79–84. ACM, 2012.

[12] Franziska Roesner, Tadayoshi Kohno, and David Wetherall. Detecting and defending against third–party tracking on the web. In *NSDI*, 2012.

[13] David Wetherall, David Choffnes, B Greenstein, Seungyeop Han, Peter Hornyack, Jaeyeon Jung, Stuart Schechter, and Xiao Wang. Privacy revelations for web and mobile apps.