

COS 597D Project - Mobile vs Traditional Web Tracking (FourthPartyMobile)

Diego Perez-Botero, Marcela Melara, Christian Eubank
{diegop, melara, cge}@cs.princeton.edu

Keywords

FourthParty, Web crawling, cookies, privacy policy, mobile

1. INTRODUCTION

We automate the detection of third-party tracking mechanisms while browsing the web on a mobile device. To this end, we adopt the FourthParty¹ project's approach and instrument a popular open-source mobile browser (i.e. Firefox) to be used as an enhanced web crawler. This enables us to log realistic end-user interactions (e.g. execution of embedded scripts) as opposed to just downloading each web page's static content, as is practice for traditional web crawlers.

The mobile web crawler is the tool we use to collect valuable information in order to conduct our comparisons between the mobile and traditional third-party tracking ecosystems and their practices. We also analyze our obtained data to characterize the behavior of emulated mobile devices and assess the extent to which they can be employed as a substitute for physical devices when carrying out web privacy research. Lastly, to help contextualize the empirical part of this project, we provide a case study on privacy policies and how they relate to the web tracking mechanisms that we see in practice.

2. RELATED WORK

There have been several other projects which share some but not all characteristics with FourthPartyMobile .

Our project adopts the methodology of FourthParty, a dynamic web measurement platform used to collect information about real-time end-user interactions [6]. With the gathered data, the authors are able to study information leakage, tracking technologies and blocking tool effectiveness [6]. Nevertheless, FourthParty was designed to just be run in browsers on desktop devices, but because of the differences between desktop and mobile platforms, we needed to port the FourthParty code to support mobile web crawling for our purposes.

In the realm of mobile web crawling, Timmins et al. [8] study content of over one-million mobile international Web pages using a search-assisted crawling methodology to locate and study pages for three of the most popular mobile web formats - WML 1.0, WML 2.0/XHTML mobile profile (XHTML-MP) and compact HTML (C-HTML). While

¹<http://www.fourthparty.info>

they look at the relative characteristics of these mobile Web content formats, as well as compare them with a similar sampling of non-mobile (HTML) content [8], the main difference between this project and FourthPartyMobile is that they do not look at the dynamic content, such as Javascript and cookies, of their crawled web pages.

For the analysis of our results of our FourthPartyMobile web crawls, we refer to Ayenson and Soltani [3, 7], who found that popular websites were using "Flash cookies" as a means of persistent tracking of users, and that many sites were also using Flash technology to reinstantiate HTTP cookies deleted by a user, a process called "respawning", making tracking more resistant to users' privacy-seeking behaviors. In addition, in [3], they also examine HTML5 local storage and Cache-Cookies via ETags as a new tracking vector.

3. IMPLEMENTATION

This section explains the design decisions and the software development process for the FourthPartyMobile project.

3.1 Challenges

The design of a mobile application/web crawler poses a variety of challenges:

- Mobile devices have limited amounts of RAM, so applications should not rely on large data structures stored in main memory.
- Security permissions in mobile devices are strict, which means that writing data into persistent memory is not always an option.
- Processing power in mobile devices is limited, so computationally intensive procedures, such as parsing a web page, should be delegated to an external entity.
- Mobile network bandwidth is a limited resource, so large data transfers should be avoided.
- Battery life must be preserved as much as possible by a mobile application if it is being aimed towards the general public.

3.2 Mobile Web Crawler's Architecture

FourthPartyMobile's architecture (see Figure 1) delegates most of the computation and storage to a supporting server, reducing the mobile device's responsibilities to 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 Megabytes in size, is generated by the supporting server's side.

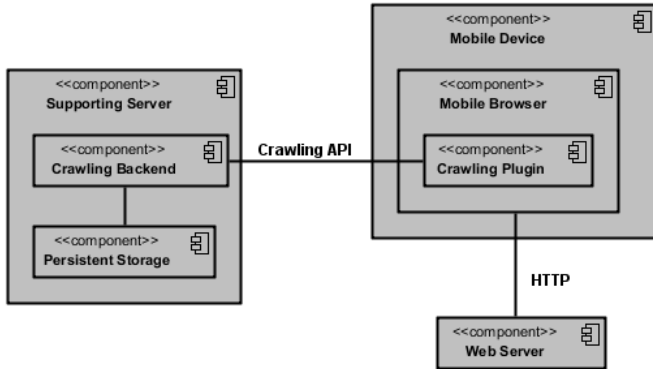


Figure 1: Prototype's Runtime Interactions.

3.3 Prototype

We took advantage of the fact that the FourthParty project is open-source. After analyzing its codebase, we ported its core functionality to support Android-based mobile devices, such as smartphones and tablets. FourthPartyMobile 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.

The FourthPartyMobile 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:** FourthPartyMobile was deployed in different devices and various test crawls were conducted to identify and eliminate all remaining bugs.

4. METHODOLOGY

This section details the steps we took to ensure that our results are highly reproducible. It also describes our testbed and the reasoning behind it. In addition, our program's source code and detailed usage instructions can be found in the project's website (<http://www.FourthPartyMobile.info>).

4.1 Automated Crawls

We tried automating crawls on Firefox Mobile with MozMill², Selenium³, Scriptish⁴ and Robocop⁵ without any success. The few frameworks that are compatible with Android do not support Firefox Mobile (fennec) – they only interact with the lackluster Android Web Browser. To the best of our knowledge (i.e. hours worth of Google searches), it seems that there are no testing frameworks out there that can automate Firefox Mobile crawls. Even the browser plugins that do this on the desktop version (e.g. Flem⁶) have not been ported to the Mozilla Mobile SDK. Bare in mind that Mozilla Mobile SDK was released on February 21 of 2012 [5], so it needs more time to mature.

Our solution was to use JavaScript code on one site to trigger the crawl in a separate tab. We tested this method on all platforms (i.e. desktop, tablet, smartphone) with positive results. Thus, we automated the generation of the HTML website containing the JavaScript code to facilitate the creation of these scripts for arbitrary URL lists. A timeout of 30 seconds was used for every one of our crawls, meaning that a new URL was visited every 30 seconds.

4.2 URL Datasets

We crawled two URL datasets parsed from *Alexa - Top Sites in United States* on January 7, 2013: (1) Top 100 and (2) Top 500. 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.

4.3 Devices

All crawls were conducted between January 7, 2013 and January 10, 2013. Six different devices were used to go through the two URL datasets, for a total of 12 result sets. We used 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)
- Emulated Nexus 7 (7.0-inch Tablet)
- HTC Evo 4G (4.8-inch Smartphone)
- 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

²<https://developer.mozilla.org/en-US/docs/Mozmill>

³<http://seleniumhq.org/>

⁴<http://scriptish.org/>

⁵<https://quality.mozilla.org/browser-technologies/mobile-firefox/robocop/>

⁶<https://addons.mozilla.org/en-US/firefox/addon/flem/>

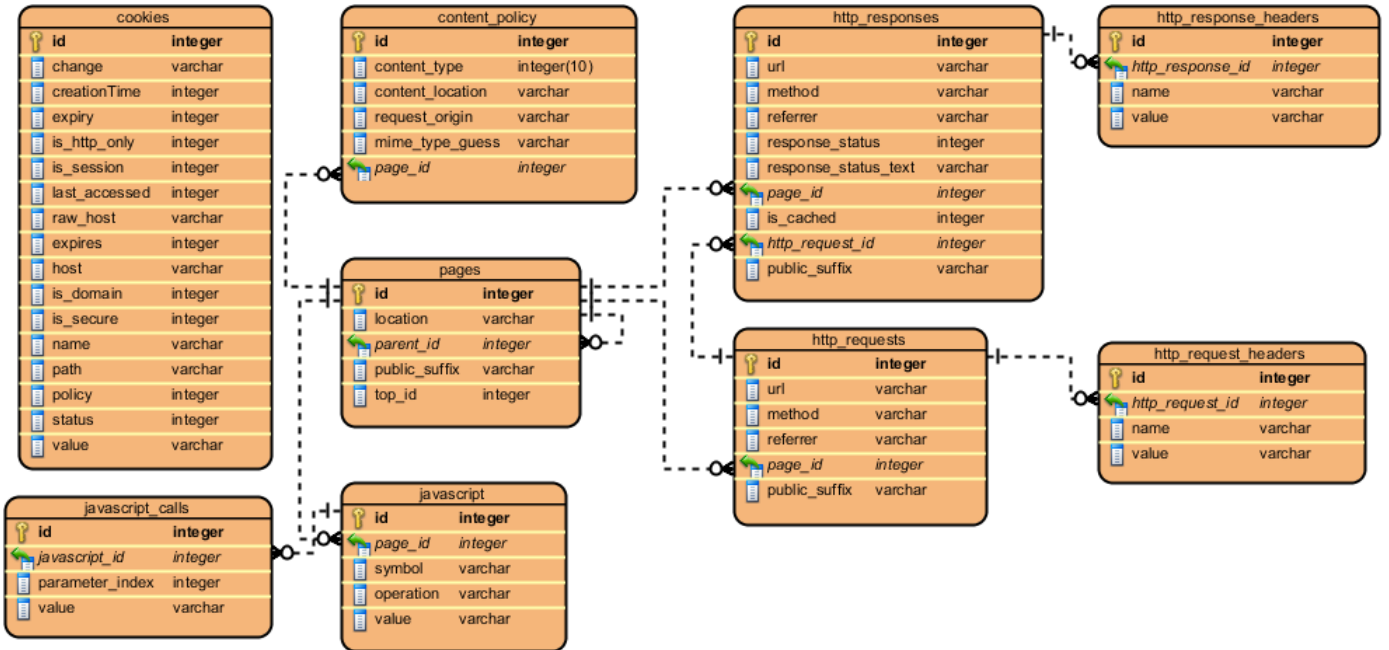


Figure 2: FourthParty SQLite database schema

SDK. We tweaked 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.

4.4 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.

Unfortunately, the mobile-enabled Mozilla Add-On SDK currently lacks support for the types of hooks required to properly intercept JavaScript calls and *content_policy* invocations. For this reason, the SQLite tables specifically allocated for JavaScript traces and *content_policy* logs are never populated during our crawls. However, HTTP responses state their content type in the headers and all JavaScript

code is transferred to the browser through HTTP responses. Thus, a simple query like the one shown in Listing 1 reveals the HTTP responses associated with that sort of code. By leveraging the power of SQL queries, we were able to carry out very detailed analyses related to JavaScript code in spite of the absence of the *javascript* and *javascript_calls* tables (refer to Section 5).

```

1 SELECT *
2 FROM http_response_headers
3 WHERE UPPER(name)='CONTENT-TYPE' AND value
   LIKE '%javascript%'

```

Listing 1: Example JavaScript analysis based on HTTP Responses

5. DATA ANALYSIS

Throughout this section, we investigate the current state of mobile web tracking based on the data collected by way of FourthPartyMobile crawls. The result sets analyzed in this section can be downloaded from the project’s website.

5.1 Main Players

In determining the sites with the most prevalent tracking habits, we first considered those sites who added the most cookies during the crawl. Specifically, we considered those sites who were in (or tied with) the top 25 for each device in terms of the number of cookies added in the 500-site crawl. Notably, third-party advertising and analytics sites comprised a sizable minority of the top 25 cookie-adders, forming 34.6% of the top 25 for desktops, 39.3% for tablets and 30.8% for phones. The majority of first-party cookie adders were online retailers and news sites.

As recorded in Table 1, roughly one third of the top cookie-adders were ranked highly across all three types of devices.

Differences in the top cookie-adders across the three devices were partially a relic of third-party websites that added cookies heavily on one platform, but not the others. Interestingly, tablets appeared to lie in between phones and computers in terms of top advertising/analytics cookie-adders. Specifically, among others, the tablet shared `2o7.net` and `collective-media.net` with phones and `dotomi.com` and `rflhub.com` with desktops. In contrast, phones and desktops did not share a top analytics/advertiser in their top 25 that did not also appear in the top 25 for tablets.

Table 1: Cookie-adding websites in top 25 for all devices

Website	Description
<code>cafepress.com</code>	Online retailer
<code>go.com</code>	Disney-owned web portal
<code>homedepot.com</code>	Online retailer
<code>hootsuite.com</code>	Social media management site
<code>pubmatic.com</code>	Online advertising company
<code>revsci.net</code>	Shell for advertiser Audience Science
<code>rubiconproject.com</code>	Online advertising company
<code>verizon.com</code>	Telecommunications/ online retailer
<code>webs.com</code>	Web hosting services

The next metric for tracking that we considered was the number of JavaScript calls made by one site that refer to another site. In most cases, these calls were to third-party advertisers and analytic organizations. These organizations were largely completely separate third-parties, but in cases such as `washingtonpost.com` and `wapolabs.com`, the first and “third” parties were actually part of the same umbrella organization.

Websites in the top 25 for containing JavaScript calls by other websites for all three types of devices are contained Table 2. Observe that these sites are predominately news and retailing sites, a trend that held for the websites not included in the table. However, beyond these 6 sites, the three devices only shared a small handful of sites in their JavaScript top 25, unlike in the case of cookie-adders.

Overall, based on our cookie and JavaScript analysis, news media and online retailers appear to be the most pervasive first-party trackers while third-party advertisers and analytics groups have a notable presence on all three types of devices.

Table 2: Websites in Top 25 for all devices for containing JS referring to other sites

Referrer	Description
<code>cbslocal.com</code>	Online news site
<code>drudgereport.com</code>	Online news site
<code>mypoints.com</code>	Online reward-points retailer
<code>theblaze.com</code>	Online news site
<code>time.com</code>	Online news site
<code>verizonwireless.com</code>	Telecommunications/ online retailer

5.2 Desktop vs Mobile Tracking

As when examining the major players in tracking, we compared desktop and mobile devices with respect to an investigation of the cookies and JavaScript calls observed during the crawls. Tables 3 and 4 contain the raw number of cookie-related calls for the devices. Unsurprisingly, as the devices became smaller and more mobile, the number of cookies

added during the crawls decreased, likely due to space constraints. However, when considering the proportion of cookies left on a device after the crawls, cookies appear to be stickier in the short term on the more mobile devices. This result is likely due to the fact that, since trackers can add a smaller number of cookies on a mobile device, they try to keep the few cookies they place on the device as long as possible.

Another basic way to categorize cookies’ longevities is by their expiration time: some are evicted from the browser’s memory after a finite amount of time, while others are meant to stay in the browser’s memory permanently. Permanent cookies are set by Mozilla Firefox to expire after exactly $9.223372036854776 * 10^{18}$ milliseconds; temporary cookies have eviction times that are more manageable (e.g. 11 days). The results of examining the expiration dates are contained in Table 5. Note that, as with short-term cookie stickiness, the proportion of permanent cookies increases as devices become more mobile.

Table 3: Desktop vs. mobile cookie action counts with Top 100 dataset

Device	Cookies - Top 100 Dataset			
	Added	Changed	Deleted	% Left
Desktop	1732	2605	148	91.45%
Tablet (avg.)	1233.5	1922.5	107	91.32%
Phone	936	1214	18	98.08%

Table 4: Desktop vs. mobile cookie action counts with Top 500 dataset

Device	Cookies - Top 500 Dataset			
	Added	Changed	Deleted	% Left
Desktop	7355	13174	640	91.30%
Tablet (avg.)	5396.5	6845.5	262	95.15%
Phone	4387	4545	101	97.70%

Table 5: Desktop vs. mobile cookie longevity with Top 500 dataset

Device	Cookies - Top 500 Dataset			
	Perm.	Temp.	% Perm.	Total
Desktop	4962	16527	23.09%	21489
Tablet (avg.)	3571.5	8932.5	28.56%	12504
Phone	2805	6228	31.05%	9033

Next, we turn to the number of JavaScript calls, including the third-party JavaScript calls. We define third-party JavaScript calls as in the previous section. Observe, in Tables 6 and 7 that the raw number of JavaScript calls as well as the proportion of calls by third-party entities fall from desktop to tablet to mobile phone. This suggests, from a tracking point of view, that JavaScript is increasingly used for purposes more in accordance with site mechanics (e.g. rendering) as opposed to tracking as devices move in the mobile direction. This could be a relic of the fact that, since these mobile devices lack the same computing power as desktops, adding extra JavaScript for purposes other than running the first-party content would slow browsing down to an unacceptable degree. Hence, one could hypothesize that, in general, trackers will use cookies (which do not require significant computational power) more frequently when compared to JavaScript on mobile devices.

Table 6: JavaScript call counts with Top 100 dataset

Device	JavaScript Calls - Top 100 Dataset		
	Total	3rd Party	% 3rd Party
Desktop	1763	1238	70.22%
Tablet (avg.)	1244.5	864	69.43%
Phone	798	454	56.89%

Table 7: JavaScript call counts with Top 500 dataset

Device	JavaScript Calls - Top 500 Dataset		
	Total	3rd Party	% 3rd Party
Desktop	10912	7310	66.99%
Tablet (avg.)	6165	3848	62.42%
Phone	4053	2344	57.83%

To test this hypothesis, in Table 8 and Table 9, we compared the number of cookie calls against the number of JavaScript calls across the different devices. Here, we defined the number of cookie calls as the sum of the number of cookies added during the crawl plus the number of cookie changes. Observe that the proportion of cookie calls hovers in the high 60’s and low 70’s percentages. Similarly, notice that this proportion is slightly but noticeably higher for mobile phones when compared to tablets and tablets when compared to desktops.

Table 8: Cookie vs. JavaScript calls with Top 100 dataset

Device	Types of Calls - Top 100 Dataset		
	Cookie	JavaScript	% Cookie
Desktop	4337	1763	71.10%
Tablet (avg.)	3156	1244.5	71.72%
Phone	2150	798	72.94%

Overall, the raw number of cookie and JavaScript calls decrease as devices become more mobile, an extremely intuitive result simply due to the fact that mobile devices have greater hardware constraints. However, the tracking behavior across devices follows a less-obvious trend. Specifically, on mobile devices, cookies appear to be designed for a greater degree of longevity and JavaScript for analytics, and in general, appears to have a lesser role. This appears to represent a greater dependency on tracking through device memory rather than through processes that requires device computation. While a user will almost certainly notice slower browsing on an already comparatively-slow mobile device, he will likely not notice the corresponding increase in memory usage for tracking purposes.

5.3 Physical vs Emulated Devices

We ran into multiple unforeseeable device-specific problems while conducting our experiments on real phones and tablets. Some Android devices did not allow applications from *Unknown Sources* to create directories inside their memory card, which was necessary for the FourthPartyMobile add-on to work. Meanwhile, others had no option to disable their inactivity timeout, which led to their screen turning off and a full reload of the crawling scripts if we did not tap the screen every 30 minutes. In some cases, incoming calls and emails, calendar notifications, and other everyday occurrences would interfere with the crawls. Even though some of the issues were solved by *rooting*⁷ the device, the time

⁷Equivalent to *jailbreaking* in iOS; unlocks privileged control over the Android OS.

Table 9: Cookie vs. JavaScript calls with Top 500 dataset

Device	Types of Calls - Top 500 Dataset		
	Cookie	JavaScript	% Cookie
Desktop	20529	10912	65.29%
Tablet (avg.)	12242	6165	66.58%
Phone	8932	4053	68.79%

that it took to set up a new testbed became unpredictable and some problems had no apparent solution. Therefore, in this section we investigate whether or not emulated Android devices are a good substitute to their physical counterparts when studying dynamic web interactions, as they provide several advantages:

- **Stability:** Emulated devices lie in the software layer and are programmed to behave in accordance with API specifications, whereas physical devices must translate the expected API-compliant behavior into hardware-dependent operations. Consequently, physical devices are more prone to error and may exhibit hardware-specific behavior.
- **Developer-Friendly:** Emulated devices are completely unlocked and expose all of the developer options that an Android Dev Phone (ADP) comes with (e.g. SIM-unlocked and hardware-unlocked).
- **Vast Resources:** Emulated devices can be assigned arbitrary amounts of RAM and persistent memory. Most importantly, they run on desktop CPUs, which are significantly more powerful than those found in a real mobile device. As a result, script execution and image rendering is less likely to overwhelm the device, which is a common issue in physical devices.
- **Easier to Attain:** Android market share is 72% as of November 2012,⁸ but that does not necessarily mean that researchers will be able to find (or afford) a good variety of Android devices for them to experiment on. On the other hand, multiple emulated devices can be instantiated on any modern PC.
- **No Background Interference:** The primary use of physical devices being used to run web crawls is not necessarily research. Incoming calls and other invasive everyday functionalities can put an ongoing crawl to an end. Meanwhile, emulated devices are dedicated to the experiment at hand and do not have any of the standard services (e.g. calendar, contacts, email) configured.

In order to assess the viability of emulated devices for future research, we will analyze the accuracy of the interactions recorded when running FourthPartyMobile on them. Ideally, an emulated mobile device (i.e. phone or tablet) should mimic its physical counterpart in such a way that: (1) the content served to it by websites is the same as the one served to its physical equivalent and (2) the content received triggers the same behavior that it would on its physical analogue. Thus, an emulated device must be able to

⁸<http://mashable.com/2012/11/14/android-72-percent/>

trick HTTP servers into thinking that it is indeed a mobile device and its dynamic properties should be consistent with those of the device being impersonated.

Given that our research is focused on privacy and web tracking techniques, cookies and JavaScript are two important mechanisms that we need to record as accurately as possible, considering that they are the most common means used by websites to store information on the client’s side and to analyze the client’s environment (e.g. installed plugins, screen size, system fonts). For this reason, we will compare cookie and JavaScript interactions seen in physical and emulated devices. At the same time, HTTP Responses will be used to determine if the actual content served to the emulated devices is the desired one or not. Throughout this section, we will be comparing the HTC Evo 4G with an emulated Nexus S smartphone, and the two physical tablets (Asus Pad TF300T and Samsung Galaxy Tab 2) with an emulated Nexus 7.

5.3.1 Cookie Interactions

There are three possible actions that can be taken when it comes to cookies: creation, modification and deletion. Tables 10 and 11 show the number of cookie events detected by our five mobile devices when crawling the Top 100 and Top 500 URL datasets respectively. It can be seen that the HTC Evo 4G and the Emulated Nexus S presented congruent numbers, while the three tablets appear to be remarkably dissimilar from one other.

Both datasets show the aforementioned trends, which are quantified in Table 12, where the differences between the emulated and expected (physical) results are presented. In the case of tablets, the two physical devices were averaged and taken as the expected result. We found that the emulated phone only distanced itself from the real phone by 16.67% or less in terms of the cookie action counts. The emulated tablet showed lower fidelity, being off by 55.73% in some cases.

Table 10: Cookie action counts with Top 100 dataset

Device	Cookies - Top 100 Dataset		
	Added	Changed	Deleted
HTC Evo 4G	936	1214	18
Emulated Nexus S	929	1130	15
Asus Pad TF300T	1093	1351	40
Samsung G. Tab 2	1374	2494	107
Emulated Nexus 7	1282	2029	97

Table 11: Cookie action counts with Top 500 dataset

Device	Cookies - Top 500 Dataset		
	Added	Changed	Deleted
HTC Evo 4G	4387	4545	101
Emulated Nexus S	4665	4946	100
Asus Pad TF300T	5163	6190	205
Samsung G. Tab 2	5630	7501	319
Emulated Nexus 7	4501	5107	116

Besides having the same amount of cookie events as a physical device, it is important that an emulated device is exposed to cookies of similar nature. As explained in Section 5.2, a basic way to categorize cookies is by their expiration time. Table 13 gives the total count of permanent and temporary

Table 12: Emulated vs Physical deltas in terms of cookie action counts

Device	Dataset	Cookies		
		Added	Changed	Deleted
Phone	Top 100	-0.75%	-6.92%	-16.67%
	Top 500	6.34%	8.82%	-0.99%
Tablet	Top 100	3.93%	5.54%	31.97%
	Top 500	-16.59%	-25.40%	-55.73%

cookies found in crawls with the Top 500 dataset, as well as their relative proportions. Once more, the Emulated Nexus S mirrors the HTC Evo 4G’s behavior fairly closely. To our surprise, the counts and nature of cookies reported by the Emulated Nexus 7 are almost the same as those of the Emulated Nexus S. As a result, the Emulated Nexus 7 is not a good replacement for either of the tablets under this criteria. Referring back to Table 11, we soon realized that the Emulated Nexus 7 has consistently been acting more like a phone than as a tablet when it comes to cookies.

Table 13: Cookie longevity with Top 500 dataset

Device	Cookies - Top 500 Dataset			
	Perm.	Temp.	% Perm.	Total
HTC Evo 4G	2805	6228	31.05%	9033
Emulated Nexus S	3002	6709	30.91%	9711
Asus Pad TF300T	3364	8194	29.11%	11558
Samsung G. Tab 2	3779	9671	28.10%	13450
Emulated Nexus 7	3014	6710	31.00%	9724

Now that we analyzed cookies by their counts and their longevity, we can dig deeper and investigate their content. Cookie size is a good metric to use when looking at the overall trends, since it gives an idea of the cookie usage patterns employed by the content providers. While some websites can write a short ID to a user’s cookie and maintain the user’s profile on their back-end, other websites might want to use cookies as a side-channel and keep all relevant tracking information on the client’s side, appending data to existing cookies.

Table 14 gives the average size of the cookies detected during the Top 500 crawls by the different mobile devices. Note that tablets reported a larger cookie size on average than phones. To improve our understanding of the cookie size distributions for the two platforms, we used 2R Soft⁹ to run a single-series analysis over the cookies associated with each crawl and used goodness-of-fit tests to select the best cumulative density function (CDF) fit. An exponential distribution gave the best Anderson-Darling statistic in all cases, with the parameters given in Table 14. We plotted the five CDFs and obtained Figure 3. Tablet crawls are represented with dotted lines and phone crawls are shown as solid lines. From the CDF curves, it is clear that the Emulated Phone (Nexus S) serves as a convincing approximation to the Physical Phone (HTC Evo 4G). Meanwhile, the Emulated Tablet (Nexus 7) remains closer to the phone crawls than to the real tablet crawls. Also, it should be noted that the two physical tablets (Asus TF300T and Samsung Galaxy Tab 2) show cookie sizes that tend to be larger than the ones registered in phones.

⁹<http://www.2rsoft.tk>

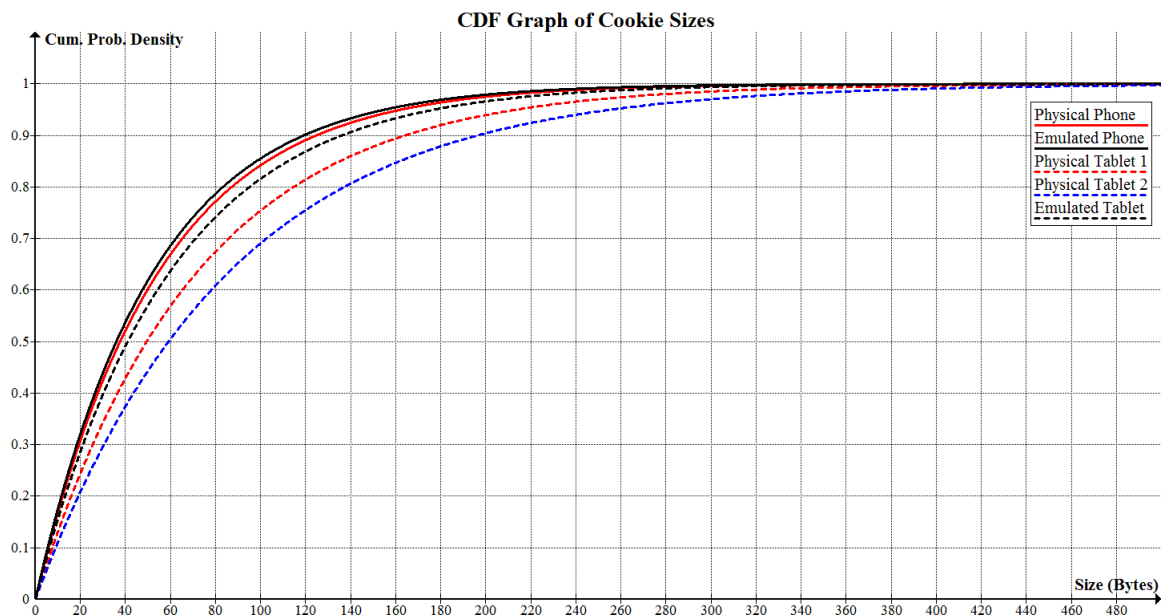


Figure 3: Cookie size distribution with Top 500 dataset

Table 14: Cookie size distribution with Top 500 dataset

Device	Cookies - Top 500 Dataset	
	Avg Size (B)	Best CDF Fit
HTC Evo 4G	54.29	Exp., $\lambda=0.01842$
Emulated Nexus S	51.86	Exp., $\lambda=0.01928$
Asus Pad TF300T	71.37	Exp., $\lambda=0.01401$
Samsung G. Tab 2	85.37	Exp., $\lambda=0.01171$
Emulated Nexus 7	59.20	Exp., $\lambda=0.01689$

5.3.2 HTTP Interactions

We are interested in comparing the content served to the various mobile devices to determine if websites are effectively fooled by the device emulator. This comes down to the HTTP Responses reported by FourthPartyMobile, which most of the time contain a *Content-Type* header indicating the MIME type of the response’s payload. From the crawl databases, we found that around half of the HTTP responses carried images in all of their formats (e.g. jpg, png, gif) and the other half was mostly comprised of text. We separate JavaScript from the other types of text-based content (e.g. HTML and plaintext) because, as mentioned before, it is an integral part of web tracking and browser fingerprinting. While we thought that Flash cookies would be of common use based on previous research [3], none of our crawls conducted on mobile devices with the Top 500 dataset contained more than 24 flash-related HTTP responses. This may be due to the fact that mobile browsers disable plug-in content by default and download such content on-demand when the user clicks on it.

Tables 15 and 16 show the counts of the various content types obtained with the Top 100 and Top 500 datasets respectively. Images are clearly the predominant type of media associated with HTTP responses in all of the crawls. At first glance, the counts reported by the Emulated Nexus S are not very distant from those of the physical phone. The same can be said about the emulated tablet (Nexus 7) with respect to

the counts of its physical counterparts (Asus Pad TF300T and Samsung Galaxy Tab 2), but only when dealing with the Top 100 dataset. The emulated tablet was less satisfactory in replicating what was seen in the physical tablets with the Top 500 dataset.

Table 17 summarizes the difference between the emulated and expected (physical) results. In the case of tablets, the two physical devices were averaged and taken as the expected result. Similar to what was seen with cookies, the emulated phone does not deviate considerably from our reference physical phone, with deltas of less than 8.33% in all counts. While the behavior of the emulated tablet was unexpectedly close to what we desired with the Top 100 dataset, the larger dataset brought about deltas of around 20% on all counts, which might be unacceptable. This would explain why, as discussed in the previous subsection, the emulated tablet did not behave the same way as the physical tablets in terms of cookie activity. After all, it appears that the emulated tablet did not receive the same web content that the physical tablets received.

Table 15: HTTP Responses by content type, Top 100 dataset

Device	HTTP Responses - Top 100			
	total	image	JS	other
HTC Evo 4G	3710	2269	721	720
Emulated Nexus S	3546	2208	678	660
Asus Pad TF300T	4440	2708	914	818
Samsung G. Tab 2	5460	3069	1294	1097
Emulated Nexus 7	4743	2848	1007	888

5.3.3 JavaScript Interactions

A high-level picture of the JavaScript code that was executed in the mobile browsers can be drawn based on the size distribution of the scripts transmitted by the content providers. Larger scripts are likely to contain more lines of code, so they are expected to execute more powerful (or intrusive) procedures on the client-side.

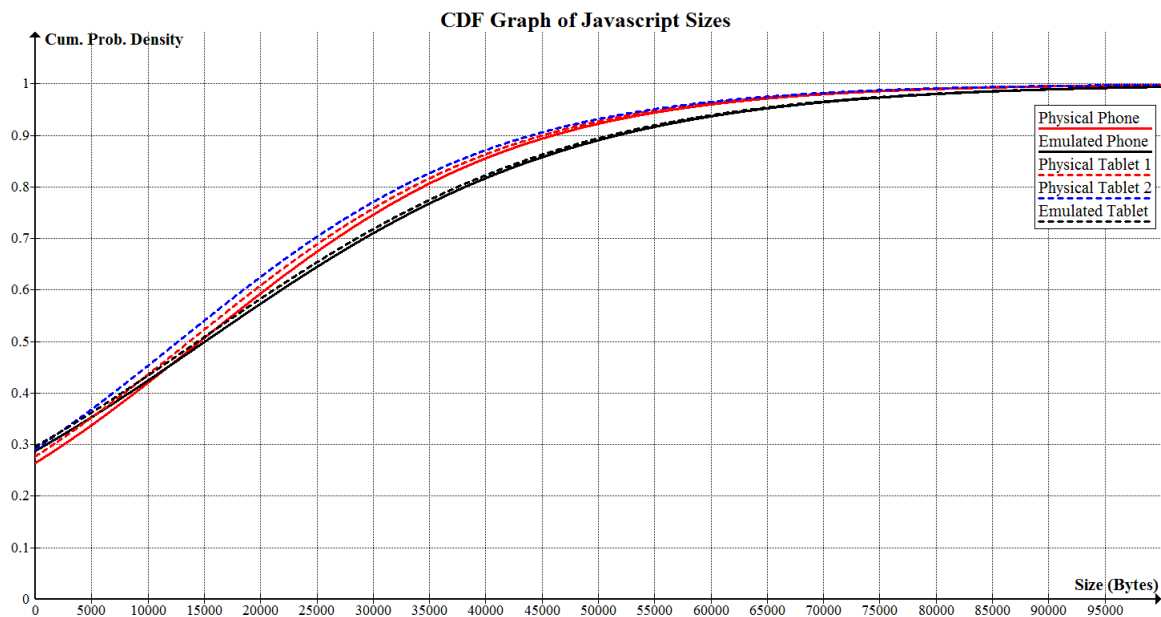


Figure 4: JavaScript script size distribution with Top 500 dataset

Table 16: HTTP Responses by content type, Top 500 dataset

Device	HTTP Responses - Top 500			
	total	image	JS	other
HTC Evo 4G	17848	10932	3616	3300
Emulated Nexus S	18376	11052	3870	3454
Asus Pad TF300T	24283	15045	5131	4107
Samsung G. Tab 2	27544	16664	6132	4748
Emulated Nexus 7	20441	12735	4298	3408

Table 17: Emulated vs Physical deltas in terms of HTTP response content types

Device	Dataset	HTTP Responses			
		total	image	JS	other
Phone	Top 100	-4.42%	-2.69%	-5.96%	-8.33%
	Top 500	2.96%	1.10%	7.02%	4.67%
Tablet	Top 100	-4.18%	-1.40%	-8.79%	-7.26%
	Top 500	-21.12%	-19.68%	-23.68%	-23.03%

Table 18 gives the average size in Bytes of the JavaScript files executed on each device during the Top 500 crawl. All averages have the same order of magnitude, so a more in-depth approach is in order. Similar to what was done with the cookie sizes, we performed a best-fit analysis with the help of 2R Soft¹⁰ and selected the most appropriate cumulative density function (CDF). Judging by the Anderson-Darling statistic, all crawls are best represented by a logistic distribution with parameters given in Table 18. The end results are plotted in Figure 4. Interestingly, the two emulated devices have JavaScript size CDF curves that are closer to each other than to that of any of the physical devices. In addition, all three physical devices have practically the same JavaScript size CDF curve. Nonetheless, the distance between the curves associated with emulated devices and the curves associated with their physical counterparts is relatively small in comparison with what was seen in the cookie

¹⁰<http://www.2rsoft.tk>

size CDFs (Figure 3) when dealing with tablets.

Table 18: JavaScript script size distribution, Top 500 dataset

Device	JavaScript - Top 500	
	Avg Size (B)	Best CDF Fit
HTC Evo 4G	14,647.11	Logistic, $\alpha=14,647.11,$ $\lambda=0.00007$
Emulated Nexus S	15,107.96	Logistic, $\alpha=15,107.96,$ $\lambda=0.00006$
Asus Pad TF300T	13,708.73	Logistic, $\alpha=13,708.73,$ $\lambda=0.00007$
Samsung G. Tab 2	12,711.00	Logistic, $\alpha=12,711.00,$ $\lambda=0.00007$
Emulated Nexus 7	14,445.22	Logistic, $\alpha=14,445.22,$ $\lambda=0.00006$

5.3.4 Our Findings

We believe that we have provided enough evidence to prove that an emulated phone (i.e. Emulated Nexus S) can impersonate a physical phone (i.e. HTC Evo 4G) with a great degree of accuracy. On the other hand, the Emulated Nexus 7 tablet consistently falls short when compared to what is expected from a physical tablet and most metrics seem to indicate that it is closer to mimicking an emulated phone than any of the physical devices.

5.4 Privacy Policy Case Study

After running our web crawls, we designed a case study in which we look at the privacy policies and related data collected from our crawls of three categories of websites that we found were amongst the most popular sites today: social networks, news sites, and e-commerce sites.

Based on the list of the Alexa Top 100 US visited websites, we chose three websites amongst the top 25 most visited sites, one representing each of our three categories. Our case study examines the privacy policies of LinkedIn¹¹ (social network), CNN¹² (news) and Amazon¹³ (e-commerce).

The case study has four stages, each stage looking at the privacy policies in more detail:

1. Compare the contents of the privacy policies displayed when visiting each of the sites on our three platforms (desktop, tablet, and smartphone).
2. Compare the length of the privacy policies of each website.
3. Examine the topics covered, i.e., the sections included in the policies.
4. Inspect some of the language used in select sections of the policies.

The question we wanted to answer in our case study was *Do privacy policies vary between normal web and mobile web?* Thus, the first stage of the case study, seeks to answer this question via a very simple method. We visited the three websites on our three platforms and downloaded the presented privacy policies for each site on each platform. This gave us three privacy policies each for LinkedIn, CNN, and Amazon.

The easiest and quickest way to determine if the contents of the policies differ for a single website was to run the `diff` command between the policies collected for each pair of platforms¹⁴: desktop - smartphone, desktop - tablet, and tablet - smartphone. Taking the more pessimistic standpoint, we were expecting to find that a single website's policies would differ more between the three platforms. But what we discovered in reality is that in the cases of LinkedIn and CNN, the privacy policies for all three platforms are exactly identical. We consider this to be a negative finding as this proves that the websites are not aware of the differences between the users' capabilities on each different platform. In the case of Amazon, the privacy policies presented when visiting the website through a desktop and a tablet are exactly identical. However, the smartphone version of the policy is much shorter and refers the reader to the full (desktop) version of their privacy notice. We would like to note that the fact that the privacy policies were basically all identical for each of the three websites in our case study greatly facilitated

¹¹<http://www.linkedin.com>

¹²<http://www.cnn.com>

¹³<http://www.amazon.com>

¹⁴The policies were all saved as text documents, and stripped of tabs and empty lines via a python script.

our further analysis of the privacy policies. The remaining examinations discussed in this section were done using the desktop version of the policies.

Our next point of study was the length of each of the privacy policies as this is an indicator of the amount of detail the websites offer their visitors. Running the `wc -w` command on each website's policy. The the resulting word counts are summarized in Table 19.

Table 19: Word counts of the privacy policies of our three studied websites.

Website	Word Count
LinkedIn	6324
CNN	2734
Amazon	2691

We were not surprised to see that the social network has the longest privacy policy, as these sites' privacy policies are often under a great deal of scrutiny. Thus, longer and more detailed is better for social networks. However, we were also expecting the e-commerce site to have a longer privacy policy, given that users purchase items through these sites by inputting sensitive data such as credit card numbers and home addresses. Thus, we were expecting to see that the website would supply the user with more detailed information about their privacy practices to reassure them that especially their sensitive data will be dealt with appropriately.

Once we had examined the privacy policies in terms of their length, we began to analyze them more carefully looking at the topics that they all cover, which are dedicated their own section in most cases. Figure 5 shows a comparison of LinkedIn, CNN, and Amazon in terms of the sections these sites include in their privacy policies at a glance. For a full list of the sections, please refer to [1, 4, 10].

As can be seen in Figure 5, there are only eight topics which all three privacy policies in our case study cover, including the presence of the last revision date at some position on the web page. It is also important to note that the sections included in CNN's and Amazon's privacy policies correspond to one of the rows in Figure 5, while several of the topics marked as covered in this figure in the LinkedIn policy are in fact subsections of more general sections of their privacy policy. For instance, Section 2 in this privacy policy covers "How LinkedIn uses your information.", where information sharing is discussed in Section 2.E and third-party content providers are explained in Section 2.K, while cookies are discussed in Section 1.G [4].

LinkedIn's privacy policy includes a few other features that the other two policies do not have. CNN and Amazon both have short introductory paragraphs in which they state the purpose and mission to protect users' privacy at the beginning of the policy. For instance, CNN's introductory paragraph states the following:

Section	Amazon	CNN	LinkedIn
What info is collected	✓	✓	✓
How info is used	✗	✓	✓
What/How info is shared	✓	✓	✓
What info choices	✓	✗	✓
How to access what info	✓	✓	✓
How to contact us	✗	✓	✓
Cookies	✓	✓	✓
Third-party ads etc.	✓	✓	✓
Security	✓	✓	✓
Safe Harbor?	✓	✗	✓
Updates to policy	✓	✓	✓
Date of last revision	top of page	bottom	top
Intro paragraph	✓	✓	✗
Summary of policy	✗	✗	✓
Intro section	✗	✗	✓
Separate cookie policy	✗	✗	✓

Figure 5: The sections covered in the policies of all three studied sites at a glance. Check-marks denote the existence of a section, X-marks denote the absence of a section.

“Thank you for visiting CNN.com, a site presented to you by Turner Broadcasting System, Inc. (“Turner”). Your privacy is important to us. As such, we provide this privacy statement explaining our online information practices and the choices you can make about the way your information is collected and used at this general audience Turner site, and among Turner’s network of affiliated websites (“Turner Network”), which includes any sites or services owned, operated or offered by or on behalf of Turner or its affiliates.” [10]

A brief statement like this serves the purpose to appease readers who may not want to read through the entire privacy policy, and who are satisfied by knowing that the website even presents such a document.

LinkedIn, on the other hand, include a short “Privacy Policy Highlights” section at the top of the web page, which summarizes the full privacy policy. This summary is great for users who do not want to go through the entire privacy policy but still want to get a general idea of the information contained in the privacy policy. Furthermore, the logo of TRUSTe¹⁵ EU Safe Harbor and a link to the TRUSTe “Click to Verify” page is displayed at the top of the privacy policy web page. This TRUSTe website seal verifies that LinkedIn’s privacy practices comply with TRUSTe’s standards [9].

Additionally, there are a few sections which we consider to be specific to a particular website category. In the case of

¹⁵TRUSTe is a global data privacy management solutions provider, part of whose mission is to ensure that digital businesses comply with international privacy requirements [9].

LinkedIn, this is Section 4 “Your obligations”, which outlines the guidelines users on the social network need to abide to in order to maintain a valid account [4]. CNN includes three extra sections that are notably related to the fact that this is a news site belonging to a larger broadcasting corporation, namely “Ad Choices”, “Turner’s Participation in the Industry Self Regulatory Program for Interest Based Ads; Additional Choice Options”, and “International Transfer” [10]. These sections mainly discuss additional practices of the broader Turner Network. Similarly to LinkedIn, Amazon’s privacy policy also includes a “Conditions of Use, Notices, and Revisions” section, in which the website not only advises users to check regularly for changes to the policy but also refers users to Amazon’s “Conditions of Use” web page and other informative pages [1].

At this stage of the case study, it became clear to us that we needed more than just to examine the sections covered in the privacy policies. Thus, the fourth phase was to investigate the language used in select sections of the privacy policies. We decided to focus on the “Cookies”, and the “Security” sections of the policies of the three websites.

One intriguing feature to examine in the “Cookies” sections is the definition of *cookie* that the websites present in their privacy policies. Table 20 summarizes the definitions found in [1, 4, 10].

Table 20: The definitions of *cookie* in the privacy policies of our three studied websites.

Website	Definition
LinkedIn	A cookie is a tiny data file that resides on your computer, mobile phone, or other device, and allows us to recognize you as a User when you return to the LinkedIn website using the same computer and web browser.
CNN	Cookies are text files placed in your computer’s browser to store your preferences. Cookies do not contain personally identifiable information; however, once you choose to furnish a site with personally identifiable information, this information may be linked to the data stored in the cookie.
Amazon	Cookies are unique identifiers that we transfer to your device to enable our systems to recognize your device and to provide features such as 1-Click purchasing, Recommended for You, personalized advertisements on other Web sites (e.g., Amazon Associates with content served by Amazon.com and Web sites using Checkout by Amazon payment service), and storage of items in your Shopping Cart between visits.

One can see that the definitions of *cookie* are driven by each website’s primary usage of cookies. Both LinkedIn and Amazon state that the main purpose of cookies is to remember a user that has once visited the website before. On the other hand, CNN simply seems to use cookies for personalization purposes, so that is the definition they offer. It is also interesting to see that the cookie itself is described in three different ways: as a “tiny data file”, as a “text file”, and also as a “unique identifier”. This, for instance, is an indication

that uniformity across all websites would be useful to avoid any kind of confusion among laypeople.

The other section we investigated in more detail was the “Security” section of all the privacy policies. Amazon and LinkedIn mention SSL encryption directly [1, 4], while CNN is very vague and simply states: “We have put in place appropriate physical, electronic, and managerial procedures to safeguard and help prevent unauthorized access, to maintain data security, and to use correctly the information we collect online” [10]. It is unclear whether or not CNN even uses any kind of encryption at all, although due to the nature of the website, it is possible that this site may not require the encryption of any information. We did not study the extent to which this may be true. However, by simply clicking on the “Log in” link, it does not seem as if this page is transmitted via HTTPS. Nonetheless, it makes sense for the other two websites to use SSL encryption since their website categories by nature require users to input and submit much more sensitive data, such as credit card numbers, and other personal information. The other issue with CNN’s language is the fact that it is overly non-technical, which may imply that the authors of the privacy policy may simply not have technical knowledge. This is another sign that it may be helpful for policymakers to receive some technical education, which would make privacy policies more uniform and informative.

In the end, we can conclude based on the first three stages of our case study that LinkedIn’s meticulously written privacy policy is the most useful for various kinds of users. Their privacy summary is an incredibly helpful tool, especially for those users who want to be informed about what their social network is doing with respect to their data and privacy, but do not or cannot take the time to read through the entire document. In addition, the fact that LinkedIn also includes a separate Cookie Policy shows that they put the time into creating documents that would fully inform their users. Another intriguing case study would be to compare LinkedIn’s privacy policy with that of other social networks to see if our conclusion about privacy policies of social networks holds.

The lessons learned in the last stage of our case study are two-fold: (1) privacy policies are clearly tailored to each of the studied website’s main purposes and goals, and (2) more consistency is required in the privacy policies across website categories to avoid confusion. While we agree that privacy policies need to reflect a website’s purpose and goals, we also argue that the extent of the discrepancies, particularly in the sections studied, may call for more uniformity across privacy policies, and indicates the potential need for more communication between policymakers and the technology community. Nevertheless, these lessons are drawn in limited premises due to time constraints, and we suggest that stage four of our case study be carried out in more detail, potentially as its own case study in the future.

We hope that our analysis of the privacy policies of the three websites chosen for our case study will serve as a starting point for further conversation on the creation of privacy policies and privacy standards.

6. CONCLUSIONS AND FUTURE WORK

By porting the FourthParty engine to mobile platforms, we have enabled future researchers to more accurately and more easily compare online tracking habits across devices. While we found that for this purpose emulated phones offer a reasonable approximation of the browsing experience, emulated tablets may provide erroneous results. We hope that researchers can leverage FourthPartyMobile to avoid this pitfall and will now view our mobile tracking analysis tool as easy-to-use as the original FourthParty for desktop. In terms of results, we found that the more mobile devices tended to use cookies with longer degrees of longevity, used cookies more heavily than JavaScript in general and had a smaller proportion of their JavaScript calls made for third-party sites.

FourthPartyMobile sets a good foundation for future privacy research in the area of mobile web tracking. While we gave a high-level view of the collected data in this paper, there is much more than can be analyzed with respect to the actual values of cookies, JavaScript calls and HTTP headers. An interesting contribution would be to train a classifier with third-party mobile cookies or third-party mobile JavaScript code from known advertisers, which can be easily obtained through the FourthParty SQLite database schema. The trained classifier could, in turn, be used to distinguish that type of content and could tell the browser to assign lower privilege levels to that content.

There are many other potential projects derived from our findings. For example, we have yet to explore other mobile platforms besides Android, such as iOS, Windows Phone and BlackBerry. In addition, we believe that the FourthParty SQLite database schema can be extended to support up-and-coming web tracking practices (e.g. stealthy *supercookies*[2]).

7. REFERENCES

- [1] Amazon.com Inc. Amazon.com Privacy Notice, Apr 2012.
- [2] J. Angwin. Latest in web tracking: Stealthy ‘supercookies’.
- [3] M. Ayenson, D. J. Wambach, A. Soltani, N. Good, and C. J. Hoofnagle. Flash cookies and privacy ii: Now with html5 and etag respawning. July 2011.
- [4] LinkedIn Corporation. Privacy Policy, Jun 2011.
- [5] D. Mason. Announcing add-on sdk 1.5!, Feb 2012.
- [6] J. R. Mayer and J. 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.
- [7] A. Soltani, S. Canty, Q. Mayo, L. Thomas, and C. J. Hoofnagle. Flash cookies and privacy. Aug 2009.
- [8] P. Timmins, S. McCormick, E. Agu, and C. Wills. Characteristics of mobile web content. In *Hot Topics in Web Systems and Technologies, 2006. HOTWEB ’06. 1st IEEE Workshop on*, pages 1–10, nov. 2006.
- [9] TRUSTe. Online privacy services - Online trust and safety.
- [10] Turner Broadcasting System Inc. CNN Privacy Statement, Feb 2011.