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

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

ABSTRACT
INSERT ABSTRACT HERE??

Categories and Subject Descriptors
H.3.5 [**Information Systems**]: Information Storage and Retrieval—*Web-based services*; K.4.1 [**Computing Milieux**]: Computers and Society—*Privacy*

General Terms
Documentation, Measurement, Security

Keywords
FourthParty, Web crawling, cookies, privacy policy, ...

1. INTRODUCTION
DIEGO

We wish to automate the detection of third-party tracking mechanisms while browsing the web on a mobile device. To this end, we will adopt the FourthParty[1] 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, which is what traditional web crawlers do.

The mobile web crawler is not our main objective for this project, but rather the tool that we will use to collect valuable information in order to conduct our comparison between the Mobile and Traditional third-party tracking ecosystems and their practices.

2. BACKGROUND AND MOTIVATION
3. RELATED WORK
MARCELA

4. IMPLEMENTATION
DIEGO

4.1 Challenges
Mobile application development poses a variety of challenges that will need to be addressed for a mobile web crawler to be materialized:

- 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.

4.2 Mobile Web Crawler's Architecture
FourthPartyMobile's architecture (see Figure 1) delegates most of the computation and storage to a supporting server, limiting the mobile deviceâĂŹs 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 a server. This way, the amount of state kept in the mobile deviceâĂŹs main memory is minimal and the crawl database, which can be several Megabytes in size, is generated by the supporting server's side.

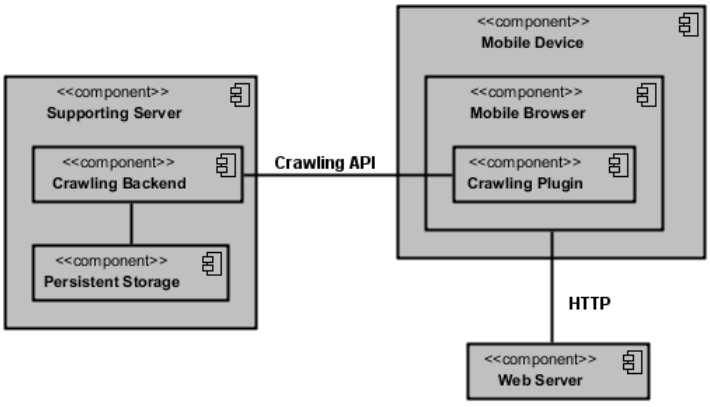

Figure 1: Prototype's Runtime Interactions.

[1]http://www.fourthparty.info

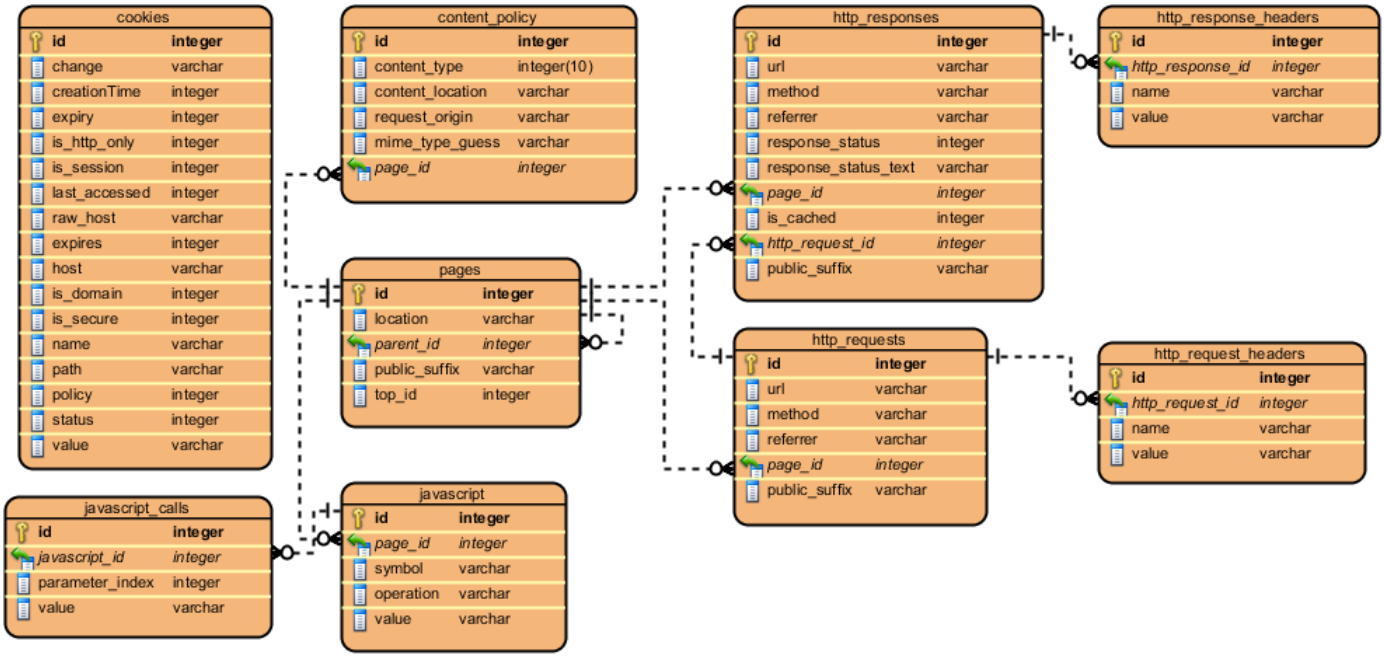

Figure 2: FourthParty SQLite database schema

4.3 Prototype

We took advantage of the fact that the FourthParty[2] project is open-source. After analyzing its codebase, we ported its core functionality over 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.

5. METHODOLOGY

DIEGO

5.1 Automated Crawls

We tried automating crawls on Firefox Mobile with MozMill, Selenium, Scriptish and Robocop without any success. The few frameworks that are compatible with Android do not support Firefox Mobile (fennec) âĂŞ they only interact with the lackluster Android Web Browser. To the best of our knowledge (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. Flem) have not been ported to the Mozilla Mobile SDK. Bare in mind that Mozilla Mobile SDK was released on February 21 of 2012 (Announcing Add-on SDK 1.5!), so it needs more time to mature.

Our solution was to use JavaScript code on one site to trigger the crawl on a separate tab. We successfully tested this method on all platforms (i.e. desktop, tablet, smartphone). This, 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.

[2]http://www.fourthparty.info

5.2 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.

After a series of crashes caused by websites containing non-western character sets, we made the decision to focus on the Top US Sites instead of the Top Global Sites.

5.3 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.

- 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)

5.4 FourthParty SQLite Database Schema

Figure 2 shows the database schema generated on every crawl.

6. DATA ANALYSIS

6.1 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 `rfihub.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
`cafepress.com`	Online retailer
`go.com`	Disney-owned web portal
`homedepot.com`	Online retailer
`hootsuite.com`	Social media management site
`pubmatic.com`	Online advertising company
`revsci.net`	Shell for advertiser Audience Science
`rubiconproject.com`	Online advertising company
`verizon.com`	Telecommunications/ online retailer
`webs.com`	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.

6.2 Cookie/JavaScript Pervasiveness

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

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

CHRIS

By website category (e.g. porn, news, etc) and by domain (e.g. com, net, etc)

6.3 Desktop vs Mobile Tracking

ALL OF US

6.4 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*[3] the device, the time that it took to set up a new test bed 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 that make them more convenient:

- **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.

[3]Equivalent to *jailbreaking* in iOS; unlocks privileged control over the Android OS.

- **Easier to Attain**: Android market share is 72% as of November 2012[4], 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 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.

6.4.1 Cookie Interactions

There are three possible actions that can be taken when it comes to cookies: creation, modification and deletion. Tables 3 and 4 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 5, 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.

[4]http://mashable.com/2012/11/14/android-72-percent/

Table 3: 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 4: 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

Table 5: 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%

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. A basic way to categorize cookies 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). Table 6 gives the total count of permanent and temporary 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 4, 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.

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.

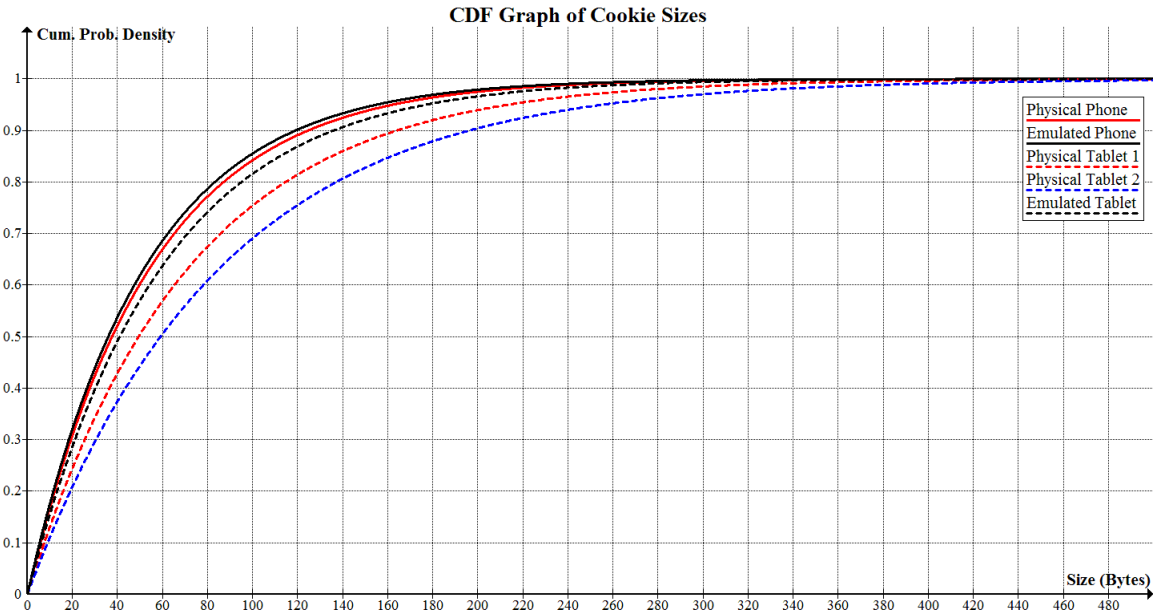

Figure 3: Cookie size distribution with Top 500 dataset

Table 6: 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

Table 7: Cookie size distribution with Top 500 dataset

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

Table 7 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 Soft[5] 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 7. 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.

6.4.2 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 a 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 [1], 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 8 and 9 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 10 summarizes the difference between the emulated

[5]http://www.2rsoft.tk

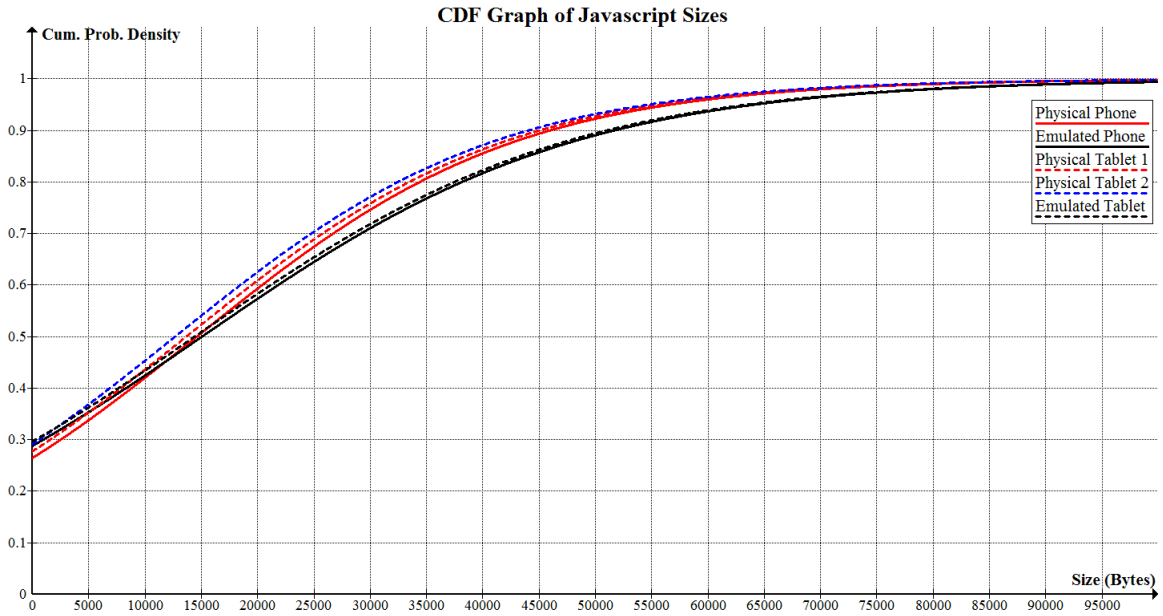

Figure 4: JavaScript script size distribution with Top 500 dataset

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 8: 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

Table 9: 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

6.4.3 *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.

Table 10: 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 11 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 Soft[6] 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 11. 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 size CDFs (Figure 3) when dealing with tablets.

6.4.4 *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

[6]http://www.2rsoft.tk

Table 11: JavaScript script size distribution, Top 500 dataset

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

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.

6.5 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. While pornography sites are also amongst the most popular sites online, we do not include this fourth category in our case study for reasons of decency.

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 LinkedIn[7] (social network), CNN[8] (news) and Amazon[9] (e-commerce).

The case study has five 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.
5. Compare the presented cookie policies with the collected web crawl data.

[7]http://www.linkedin.com
[8]http://www.cnn.com
[9]http://www.amazon.com

The first 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 platforms[10]: 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. **keep this**We consider this to be a positive finding as this proves that the websites make an effort to maintain consistency across the devices.**??** 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 our further analysis of the privacy policies. Any further examination was 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 12.

Table 12: 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 types of websites are in a constant race to see who can protect their users' privacy better. 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. What we were not expecting was to find that Amazon would have the shortest policy of our three studied websites.

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

In the end, we can say that LinkedIn's meticulously written privacy policy is the most informative 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.

7. CONCLUSIONS AND FUTURE WORK

8. REFERENCES

[1] 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.