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.

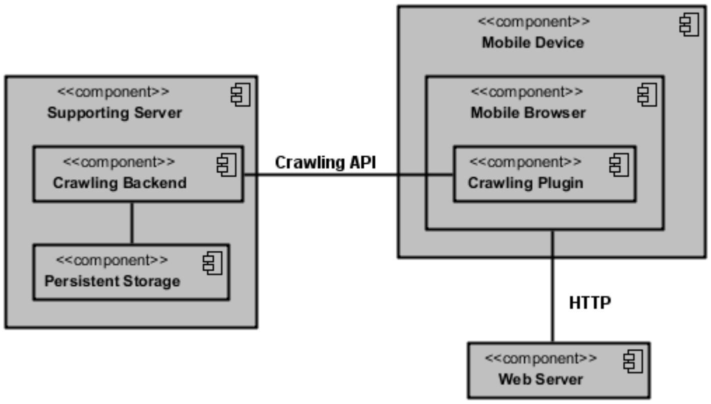

Figure 1: Prototype's Runtime Interactions.

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

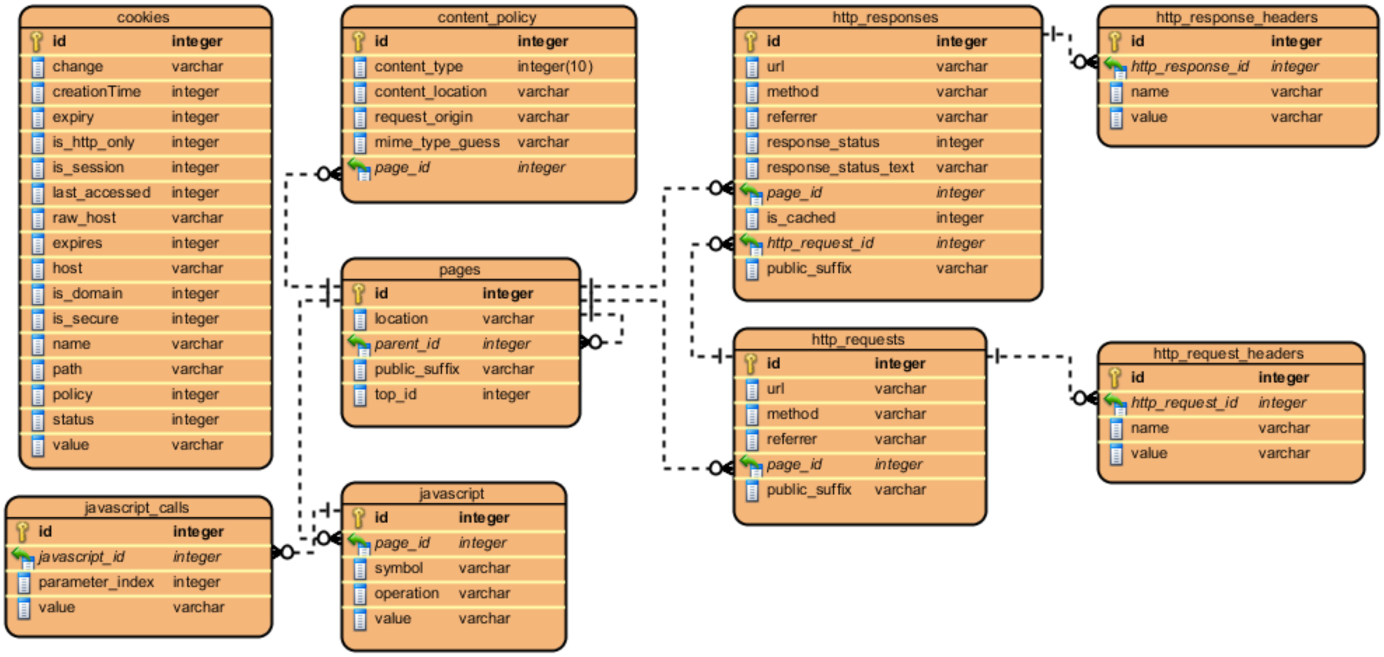

Figure 2: FourthParty SQLite database schema

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.

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.

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

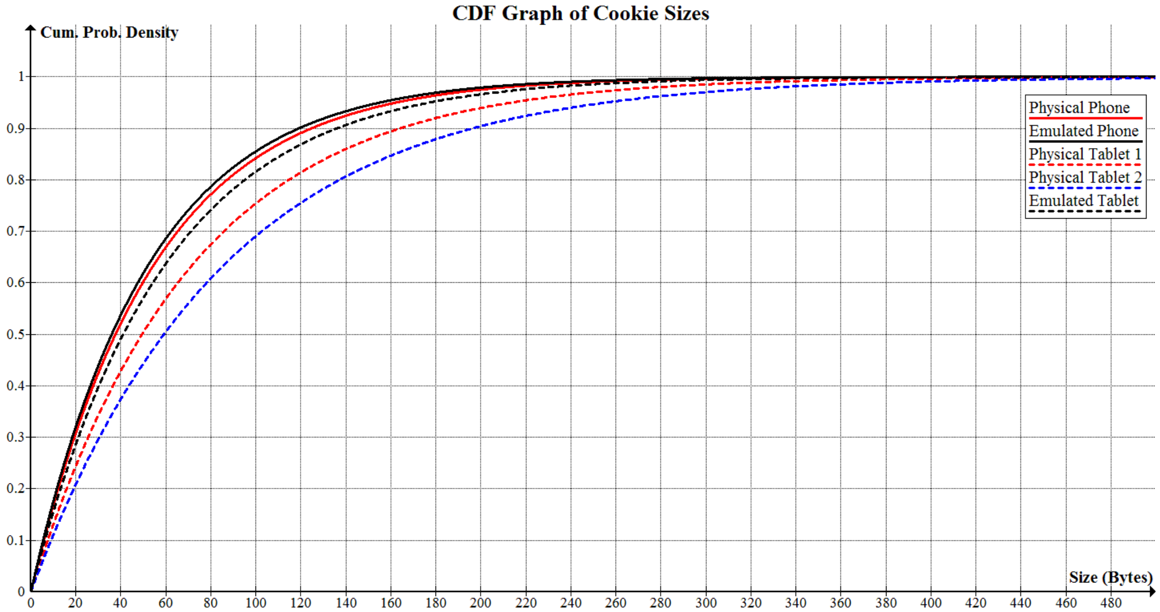

Figure 3: Cookie size distribution with Top 500 dataset

6. DATA ANALYSIS

6.1 Main Players

CHRIS

Three types of players: advertisers, content providers, and third-party content providers (embedded in sites)

6.2 Cookie/JavaScript Pervasiveness

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

DIEGO

6.4.1 *Cookie Interactions*

Table 1: 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

6.4.2 *HTTP Interactions*

6.4.3 *JavaScript Interactions*

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

Table 2: 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 3: 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%

Table 4: 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 5: Cookie size distribution with Top 500 dataset

Device	Cookies - Top 500 Dataset	
	Avg Size (Bytes)	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

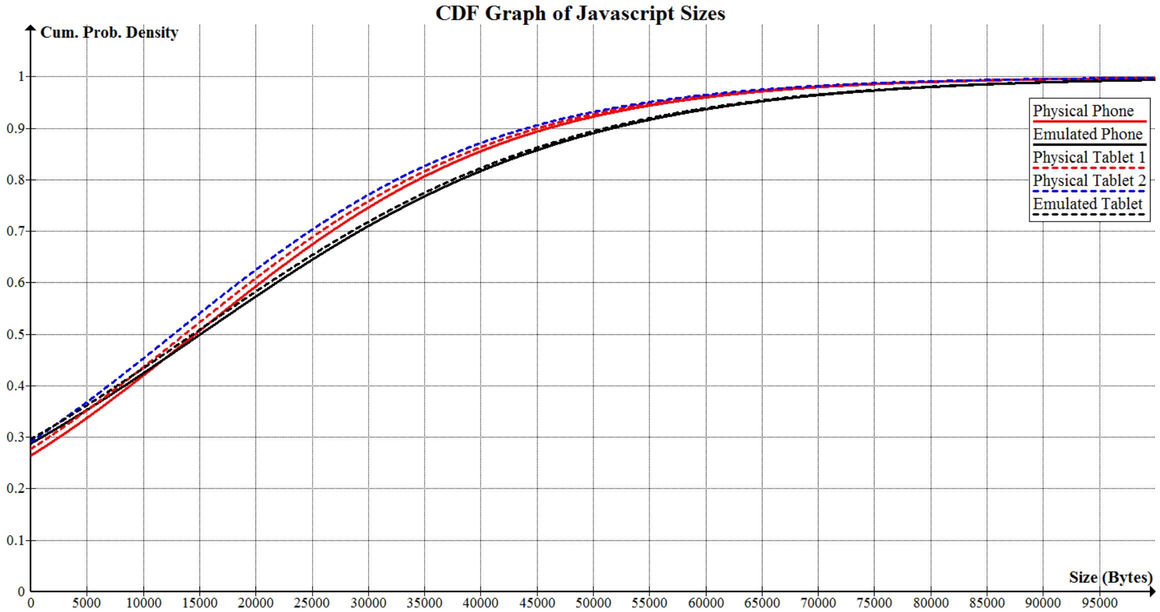

Figure 4: JavaScript script size distribution with Top 500 dataset

Table 6: 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 7: 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 8: 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 9: 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

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[3] (social network), CNN[4] (news) and Amazon[5] (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.

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[6]: 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 10.

Table 10: 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.

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

[3] http://www.linkedin.com

[4] http://www.cnn.com

[5] http://www.amazon.com

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