Should You Use the App for That?

Comparing the Privacy Implications of App- and Web-based Online Services

Christophe Leung, Jingjing Ren, David Choffnes, Christo Wilson
Northeastern University

ABSTRACT

Many popular, free online services provide cross-platform interfaces via Web browsers as well as apps on iOS and Android. To monetize these services, many additionally include tracking and advertising libraries that gather information about users with significant privacy implications. Given that the Web-based and mobile-app-based ecosystems evolve independently, an important open question is how these platforms compare with respect to user privacy.

In this paper, we conduct the first head-to-head study of 50 popular, free online services to understand which is better for privacy—Web or app? We conduct manual tests, extract personally identifiable information (PII) shared over plaintext and encrypted connections, and analyze the data to understand differences in user-data collection across platforms for the same service. While we find that all platforms expose users' data, there are still opportunities to significantly limit how much information is shared with other parties by selectively using the app or Web version of a service.

1. INTRODUCTION

Web browsers and mobile apps are the dominant media through which people interact with online services such as social media, news, weather, and dating. Many of these services are provided for free to users, with providers supporting their costs through revenue from advertising and data analytics. This necessarily raises important privacy concerns regarding what information is collected about users and how it is used.

Previous work investigates the question of what information is collected, either in the Web browsing environment [8, 15, 22, 24, 33–35] or in the mobile environment [29, 38, 42]. A close reading of this literature reveals differences between these media, with the Web having more sophisticated tracking infrastructure overall, versus apps which have more direct access to sensitive information through APIs. However, to date no work has directly compared these media for the same service to understand a fundamental question: is there a medium that is better for privacy—app or Web?

This paper provides a first look at this issue, which requires addressing two key challenges. First, we must gather a representative sample of information that large numbers of online services expose of the Internet, both via apps and Web sites. Second, we must reliably identify the personally identifiable information (PII) in network traffic generated by these services. By providing greater transparency into how apps and Web sites share PII, we seek to provide the community with better insight into the data collected by specific apps and Web sites, as well as help users make informed decisions about how they interact with online services.

To address the first challenge, we use a dataset consisting of network traces gathered from manual interactions with iOS, Android, and Web versions of the same 50 free online services. This includes major services like The Weather Channel, Yelp, and BBC News. We address the second challenge by conducting controlled experiments where ground-truth information about users' PII, coupled with state-of-the-art inference techniques to identify PII in network flows [38]. Unlike our prior work that exclusively focuses on PII leaked by apps, this paper aims to provide a comparison of Web- and app-based data collection by the same service.

Using this approach, we determine the PII exposed by services over plaintext and/or to advertising and analytics (A&A) third-party domains, and analyze their implications on privacy. Our key findings are as follows.

- **Should you use the app? It depends.** Due to the potentially large set of PII that apps can access with user permission, we expected that they would generally leak more PII than Web sites. However, we find that in 40% of cases, Web sites leak more types of information than apps. To help guide users toward using an app or Web site for a specific service, we provide an online interactive interface that makes custom suggestions based on user-specified privacy preferences at: `https://recon.meddle.mobi/appvsweb/`
- **What information leaks more from different media?** We find that locations and names leak more often from Web sites than from apps, whereas only apps leak unique identifiers and other device-specific information. Surprisingly, we find passwords leaked (albeit over HTTPS) to third parties that have no reason to receive them.
- **Web sites directly contact more trackers and advertisers than apps.** We find that Web sites often include content from multiple advertisers and third

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

IMC '16, November 14–16, Santa Monica, CA, USA.

© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ISBN 978-1-4503-4526-2/16/11…$15.00

DOI: `http://dx.doi.org/10.1145/2987443.2987456`

parties, and cause browsers to redirect through several more via real-time bidding. In contrast, most apps include a single advertisement library, which contacts fewer domains.

- **How much tracking is in common between app and Web for the same service?** We find that both apps and Web sites can leak locations, names, gender, phone number, and e-mail addresses. Unlike for apps, we found no evidence in our tests that Web sites are able to access and share device-specific unique identifiers, such as an IMEI and a MAC address. Whether this is true for other services remains an open question.

In addition to providing an online interface to make customized privacy recommendations, we make our dataset and code available at:

`https://recon.meddle.mobi/appvsweb/`

2. BACKGROUND AND RELATED WORK

Users are increasingly concerned with the amount of tracking and data collection conducted by online services [32,41]. In response, regulators such as the FTC, FCC, and the EU Commission enacted rules that protect consumer privacy; non-profits such as the Data Transparency Lab and Mozilla support efforts to increase transparency of online tracking; and tools like AdBlock and Disconnect limit tracking.

These efforts are supported by a large body of research that identifies when Personally Identifiable Information (PII) is exposed by online services. Previous work focuses either on Web sites or apps to determine privacy risks, but not both. In contrast, to the best of our knowledge, we are the first to directly compare information gathered through Web sites and apps for the same online service, allowing us to provide a relative ranking of which one is less invasive according to various metrics. Although this study represents a snapshot of online service behavior at one point in time, our approach is general and can be repeated to observe how the privacy landscape evolves.

2.1 Web Privacy

Well before there were apps and modern smartphones, researchers observed that advertisers and analytics companies were tracking users via Web site content [25]. These initial observations motivated a wide range of research on Web tracking, from understanding the tracking ecosystem over time and the economics behind it [11,18,26,27], to identifying specific techniques used to track users [5,8,15,22,24,33–35,39], to examining how tracking varies geographically [16]. While several proposals attempt to help users regain control over their privacy when browsing the Web [28,36], tracking remains pervasive.

Unlike prior work, our paper focuses on characterizing third-party tracking and the PII they collect for *services that are also available as apps*. Further, to the best of our knowledge no other study focuses on Web tracking and its privacy implications from mobile browsers. (For our purposes, only the operating system's native browser application is considered. Embedded browser components such as `WebViews` are not included.) This is an increasingly important distinction, as mobile browsers have access to sensors (e.g., GPS) that are not available on desktops.

2.2 Mobile App Privacy

Due to the rich sensors, APIs, and availability of PII on mobile devices, a large body of work focuses on understanding privacy from the perspective of tracking and data-collection by mobile apps. Early testbed studies showed that popular apps exposed location, usernames, passwords, and phone numbers [40]. Follow-up work observed similar behavior at scale "in-the-wild" [29,38,42]. A number of projects focus on detecting and mitigating privacy violations from mobile apps [6,7,12,14,17,19,21,23,30,38,43–46].

In this paper, we focus on comparing the PII exposed by mobile apps and Web sites for the same service. To accomplish this, we use tools from prior work [38] to identify PII leaks in mobile-device traffic.

2.3 Mobile Experimentation Methods

For scalability reasons, most previous work uses automated tests to analyze mobile apps [9,20,31]. However, a key limitation of this approach is that they cannot automatically explore apps that require signing in [13]. Further, our recent study shows that automated tools only reveal a small fraction of the PII exposed when manually interacting with apps [38]. In this work, we use manual tests of Web sites and apps, both to ensure that the PII exposure is representative of what users would see, and to ensure that we explore the same features of the service across both Web and app.

3. DATA COLLECTION

In this section, we describe the online services we investigated, our experimental methodology for eliciting and identifying PII sent over the network, and high-level statistics about our gathered dataset.

3.1 Selecting Online Services

Our first task is selecting online services to measure, each of which must meet the following criteria: 1) it must be popular (according to app store rankings) and/or "featured" in an app store, 2) it must provide a free app in the Google Play Store and the Apple App Store, 3) it must provide equivalent functionality via a mobile Web browser, and 4) it must not implement certificate pinning. For example, Instagram fails criteria (3) because the mobile Web site does not offer the same functionality as its app. Similarly, Pandora fails because it will not stream music via Chrome on Android. Facebook's app fails criteria (4). In general, we omitted any service for which we could not make an apples-to-apples comparison.

To locate candidate apps, we crawled the top 100 free Android apps listed in the US version of the Google Play Store on March 23, 2016. To avoid personalized recommendations that would impact the set of presented apps, we browsed the Google Play Store with a clean browsing history and no cookies stored. Only 75 apps met the requirements for our study. We added to this set "featured and recommended" apps that were promoted on the home page of the Google Play Store. In total, we selected a subset of 50 services to test, and chose them based on broadly covering popular apps across different app categories, then filling in with apps that are likely to collect PII (shopping, travel, entertainment). While we cannot make any claims about generality, we believe this set provides an interesting cross-section of online services with respect to privacy.

3.2 Experiment Methodology

Understanding privacy implications of mobile apps and Web sites requires interacting with these services in ways that normal users would. Using automated testing frameworks for this purpose is tempting, due to their simplicity, low effort, and ability to test large numbers of apps in a short period of time. However, previous work show that such tests miss important UI features (e.g., logging in, entering valid user data into text fields) [38]; further, there is a lack of good automated testing tools for iOS and for mobile browsers.

Instead, we conducted *manual* tests of 50 online services. Manual tests avoid the pitfalls of automated ones because testers can interpret UIs, enter reasonable data into arbitrary fields, and ensure similar (or identical) service functionality is exercised both over apps and Web sites. While we cannot claim generality or representativeness based on the 50 online services we tested, these comprise some of the most popular services used in the United States. We used the following procedures to test each online service.

Test Environment. Each test consisted of interacting with a given service via an app or Web site for four minutes. We collected network traffic generated during each experiment using Meddle [37], and used Mitmproxy [3] to capture both HTTP and the plaintext content of HTTPS flows. For each service requiring a login, we created a new account using a previously unused email address.

We used two phones (a Nexus 4 and a Nexus 5) running stock Android 4.4, and two phones (both iPhone 5's) running iOS 9.3.1. We specifically chose to test on Android 4.4 because it was the most common Android version in-the-wild as of April 2016 [4]. All three phones were factory reset before our experiments, and included no apps beyond the stock services and the 50 apps evaluated in this work.

Interacting with Services. Each experiment used the following steps. We installed the service's app, then connected the device to Meddle using a VPN tunnel. Next, we opened the app and used it for its intended purpose for approximately four minutes. We approved any system permission requests when prompted. After the time expired, we closed the VPN connection and uninstalled the app.

We repeated this procedure using the operating system's default browser: Chrome for Android, and Safari on iOS. To avoid contamination due to browsing history and stored cookies, we used "private mode" browsing. When interacting with the Web version of the service, we attempted to conduct identical operations as in the app (to the extent possible). To ensure fairness, when asked to log-in, we used the same pre-created account credentials used to test the app.

Note that we cannot claim to exhaustively cover all potential PII leaks using only four minutes of manual app testing. However, based on a number of tests using longer durations (10 minutes) for a subset of apps (the five apps that leaked the most and least during 4-minute tests), we found that four minutes strikes a good balance between providing adequate time to use most features of a service, and quickly covering a reasonably large number of apps in a fixed amount of time. Specifically, we found that the number of third parties contacted and number of times PII leaked were roughly proportional to the duration of the experiment (because longer experiment durations leads to more network flows), but we generally *did not see additional types of PII* leaked during the longer experiment duration (with the exception of one additional PII type, e-mail address, leaked from one app after four minutes).

Regardless, our results represent *a conservative lower bound on the PII leaked from apps and Web sites.* Based on the substantial amount of leaks discovered, we believe this to be an important first step toward understanding differences between PII leaks over apps and Web sites.

Filtering. One issue with collecting network traces from mobile devices is that flows may be generated by the foreground process (i.e., the app or Web site we are investigating) or background processes. We use three methods to minimize background traffic from our traces. First, we use a clean, factory-reset lab phone to conduct the tests. Second, we turn off background synchronization and manually close all background apps before each experiment. Finally, we filter traffic to domains that are known to be associated with OS services (e.g., Google Play Services and Apple iCloud).

Identifying PII. The next step in our methodology is identifying PII in our network traces. This task is greatly simplified because our experiments are controlled, i.e., we know all the PII that is available on our test devices. This includes usernames and passwords, MAC address, IMEI, GPS coordinates, ZIP code, *etc.*

However, knowing the PII in advance is not a catch-all for detecting it in network traffic. GPS locations are sent with arbitrary precision, unique identifiers are formatted inconsistently, a user's inferred gender is not stored in the phone, *etc.* Thus, we use the following approach to identify PII. First, we use the automated ReCon tool [38], which uses machine learning to detect likely PII in network traffic without needing to know the precise PII values. Second, to minimize the risk of ReCon missing PII, we augment its results with PII found via direct string matching on known PII. Finally, we manually verify ReCon predictions and excluded false positives based on our ground-truth information.

Domain Categorization. The final step in our methodology is labeling all the flows based on their destination. We manually identified first-party flows by looking for domain names associated with our chosen services (e.g., `weather.com` and `imwx.com` for the Weather Channel). For the remaining third-party flows, we further categorize them as advertisers or analytics by comparing the destination domain to EasyList [2].

Defining a PII "Leak." We focus on PII that reduces users' privacy either because (1) it is transmitted over the Internet unencrypted, thus exposing the data to eavesdroppers, or (2) it is sent to third parties (encrypted or plaintext) and is not required for logging into the service, thus exposing users to profiling. *We label network flows containing PII under these two conditions a PII leak.* If a username, password, or e-mail address (often used as a username) is transmitted to a first-party site[1] over HTTPS, then we *do not* consider them to be leaks. All other cases of transmitted PII are leaks. For example, a birthday sent to a first party using encryption is a leak; the same is true or an e-mail address sent in plaintext and/or to a third party.

While many first party "leaks" may be intended and acceptable to the user, we err on the side of identifying all PII sharing beyond login credentials to provide a broad view of data-collection when using online services. Such informa-

[1] Or to a single sign-on service.

tion can help users evaluate (and re-evaluate) the implications of sharing their PII over time and across services and platforms.

Experiment Limitations. Our experiments are limited to detecting PII leaks that occur directly to first and third parties, and that are detectable using common encodings (i.e., are not obfuscated). Identifying cases of users' PII shared by other parties indirectly is an important topic of research beyond the scope of this short paper. We were not able to measure services that use TLS certificate pinning, such as Facebook and Twitter, because they prevent us from decrypting network traffic with Meddle.

We found no evidence of PII leaks from browsers themselves, or from apps to browsers (or vice versa). However, this was by design and is a limitation of our work. In this paper, we are primarily concerned with the PII that apps and Web sites directly gather from users. To achieve this, we took several steps to eliminate leakages across media, including: using factory-reset OSes and their respective default browsers for each session; using private mode to browse, and different credentials for each test. Properly identifying browser (or cross-site) leaks is an open and challenging question, one that is outside the scope of this short paper.

3.3 Dataset

We manually tested online services over app and Web versions in the Boston area between March 23 and May 11, 2016. Table 1 summarizes the services that leaked PII by OS, medium (app vs. Web), and by category. In addition to the number of services tested under each OS and service category (first column), we show the average popularity rank of the apps we tested (second column) using data from App Annie [1]. We observe that most apps are within the top-40 for their category. We will discuss the information exposed by these services (third and fourth columns) in Section 4.2.

4. RESULTS

This section summarizes our key findings with respect to the privacy implications of using apps or Web sites for online services. We first focus on requests to third-parties, then analyze the PII exposed by these services, and finally conclude with how effectively online services can track users across app and Web platforms.

4.1 Third-Parties

In this section, we focus on the third-parties that are contacted by online services. Specifically, we focus on *advertising and analytics (A&A)* domains, because it is well-known that they track users in order to serve targeted ads.

Figure 1a depicts a CDF of the difference in the number of unique A&A domains contacted by app- and Web-based versions of the each online service. We present one curve for each OS. Negative values indicate that the Web version of the service contacts more domains than the app version.

Figure 1a shows that the vast majority (83% on Android, 78% on iOS) of online services contact more third-parties via their Web site than their app. Some of the greatest disparities come from services like Accuweather, BBC News, and Starbucks, which contact ≤ 4 third-parties in-app, but contact tens of A&A domains on their Web sites.

A&A domains are also responsible for the different amounts of network traffic required to use the service. Figure 1b shows a CDF of the difference in the number of network flows between app- and Web-based versions of each online service. The key takeaway is that the inclusion of additional A&A sites in Web versions of a service are often responsible (for 73% of Android services and 80% of iOS) for hundreds and sometimes thousands of extra TCP connections. Services that trigger over thousands of TCP connections include All Recipes Dinner Spinner, BBC News and CNN News, over the course of four-minute interactions in our experiments. These connections can further be wasteful in terms of bandwidth, sometimes leading to several MB of data consumption during only 4 minutes of interaction time (see Figure 1c).

To summarize, based on the pervasiveness of direct tracking from A&A sites, we find it is nearly always better to use an app than a Web version of a service. In the next section, we include PII leak information to better understand how much information is exposed by each service.

4.2 PII Leaks

This section focuses on what PII is leaked, how this differs between app- and Web-based versions of services, which third-parties receive leaked PII, and the amount of overlap between PII leaked from apps and Web sites.

Aggregate View. We begin with PII leaks aggregated by platform and category (second and third column groups in Table 1). The second column group shows the fraction of services that leak PII, and the average number of domains receiving PII leaks per service.

A few clear trends emerge. First, we observe that 14% more services leak PII via apps than via Web sites (first two rows), though the overall fraction of leaky services is high in both cases. Next, we see that while similar fractions of Android and iOS apps leak PII, 24% fewer Web sites leak PII when loaded in Chrome on Android vs. Safari on iOS. However, we also see that Web sites leak comparable types of PII regardless of whether they are loaded in Chrome or Safari (with phone number being the sole exception).

When grouping services by category, we find that apps leak an equal or greater amount of PII compared to the corresponding Web sites. The categories leaking PII to the most domains are *Education* and *Weather*, while *Entertainment* (which is dominated by streaming video apps) is least likely to leak.

Focusing now on the leaked identifiers in the last column group in Table 1, we find that every category leaks unique identifiers (column **UID**), and almost all Web and apps leak location (column **L**, either GPS coordinate or ZIP code). Some services leak gender and birthdays, even though that is not something entered by the user during tests (they were entered at account creation before testing).

Importantly, we found four cases of *password leaks* to third parties over HTTPS connections. Specifically, we found that Grubhub sent passwords to `taplytics.com`, JetBlue to `usablenet.com`, and The Food Network and NCAA Sports sent passwords to Gigya, a third-party identity management service.

We reported the first two cases to Grubhub and JetBlue, respectively, according to responsible disclosure principles.[2]

[2]We did not report the Gigya cases because they were clearly intentional behavior and not a security vulnerability per se, even though users were likely unaware that a third-party credential-management service was used.

			# of Services	Avg. Rank	PII Leaks: Services	PII Leaks: Domains	Leaked Identifiers: B	D	E	G	L	N	P#	U	PW	UID
	All	App	50	32.6	92.0%	4.7 ± 4.7	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
		Web	50	-	78.0%	3.5 ± 3.1	✓		✓	✓	✓	✓	✓	✓	✓	
OS	Android	App	48	35.4	85.4%	2.4 ± 3.4			✓	✓	✓	✓			✓	✓
		Web	48	-	52.1%	2.6 ± 2.8	✓		✓	✓	✓	✓		✓	✓	
	iOS	App	50	30.1	86.0%	4.1 ± 4.4	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
		Web	50	-	76.0%	3.1 ± 2.8	✓		✓	✓	✓	✓	✓	✓	✓	
Category	Business	App	2	3.0	100.0%	3.0 ± 0.0					✓					✓
		Web	2	-	50.0%	3.0 ± 0.0					✓					
	Education	App	4	16.1	75.0%	11.7 ± 14.4		✓		✓	✓					✓
		Web	4	-	50.0%	2.0 ± 1.0				✓	✓			✓		
	Entertainment	App	6	16.3	66.7%	6.0 ± 2.5					✓			✓		✓
		Web	6	-	50.0%	1.3 ± 0.5			✓	✓	✓	✓			✓	
	Lifestyle	App	6	57.9	100.0%	4.2 ± 2.3		✓	✓		✓	✓	✓		✓	✓
		Web	6	-	100.0%	4.5 ± 3.4			✓	✓	✓	✓	✓		✓	
	Music	App	4	92.4	100.0%	2.8 ± 2.0		✓	✓		✓				✓	✓
		Web	4	-	50.0%	4.5 ± 1.5			✓					✓	✓	
	News	App	2	4.0	100.0%	4.5 ± 3.5		✓		✓	✓					✓
		Web	2	-	100.0%	3.0 ± 0.0				✓	✓					
	Shopping	App	9	13.7	100.0%	3.3 ± 0.9		✓	✓		✓	✓	✓		✓	✓
		Web	9	-	77.8%	4.3 ± 4.2				✓	✓	✓		✓		
	Social	App	2	24.2	100.0%	6.0 ± 0.0		✓	✓	✓	✓					✓
		Web	2	-	100.0%	1.5 ± 0.5			✓	✓	✓					
	Travel	App	12	47.2	91.7%	3.7 ± 1.3	✓	✓	✓		✓	✓	✓	✓	✓	✓
		Web	12	-	91.7%	3.1 ± 3.0	✓		✓	✓	✓	✓	✓	✓		
	Weather	App	3	3.3	100.0%	8.3 ± 2.1				✓	✓					✓
		Web	3	-	100.0%	5.7 ± 3.3					✓					

Table 1: Summary of tested services, broken down by OS and category. *The vast majority of services leak PII, with apps leaking more frequently than the corresponding Web site. The leaked identifiers are* **B***irthday,* **D***evice Info,* **E***mail address,* **G***ender,* **L***ocation,* **N***ame,* **P***hone #,* **U***sername,* **P***ass***W***ord, and* **U***nique* **ID***entifiers.*

Grubhub confirmed that the passwords were inadvertently sent via an encrypted connection to tapltyics.com, Grubhub's analytics provider. Grubhub confirmed it was a bug and released a new version of the app addressing this bug within a week after confirmation, and confirmed deletion of all data by `taplytics.com` that was sent in error.

JetBlue informed us that the password was intentionally sent to `usablenet.com` for authentication services, and that in addition to using encryption to send the password over the network, it is also encrypted before storing.[3] In The Food Network and NCAA Sports cases, an important issue is that users are not made aware that their credentials are managed by another party, since the login pages are hosted by the first party site and do not mention the third party.

Following the rows in Table 1, we find that Shopping and Travel services leak the widest variety of PII, including phone numbers, as well as usernames and passwords to third-parties (via HTTPS). On the other hand, Business and Weather apps leak the fewest types of PII.

In summary, we find that PII leaks are pervasive and differ according to app category. In general, apps leak more PII than Web sites, which is expected since apps can request direct access to more types of PII stored on the device than a Web site. Interestingly, Education and Weather services are both the most promiscuous at leaking PII (contacting the largest number of domains) but leak fewer types of PII than other categories.

Differences in PII Leaks. We now focus on how app- and Web-based versions of the same service differ in terms of PII leaks. We analyze the number of domains receiving leaks, the number of distinct identifiers leaked, and the overlap in leaked identifiers.

Figure 1d shows a CDF of the difference in number of domains receiving PII leaks between app- and Web-based versions of the each online service, with negative numbers indicating the Web site leaked PII to more domains. We observe very different trends compared to A&A domains shown in Figure 1a. The curves show that there is a slight bias toward apps leaking PII to more domains than Web sites.

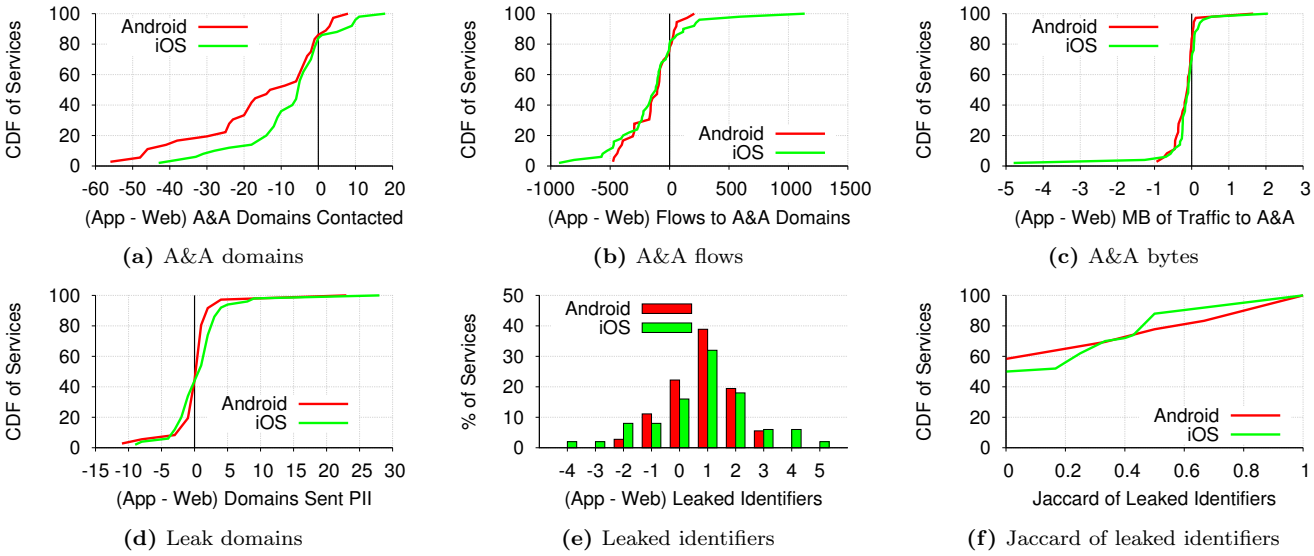
To understand *how many* distinct types of PII are leaked, we plot a PDF of the difference in leaked identifiers for the app- and Web-based version of the same service (Figure 1e). The figure shows that the most common case is that both the app version of the service leaks one more type of distinct PII than the Web site, and there is a strong bias toward apps leaking more distinct types of PII than Web sites (positive x-values).

A key question is whether app- and Web-based versions of services are leaking the same set of PII or not. We analyze this using the Jaccard index, which is a metric of set similarity where 0 means nothing in common and 1 means the sets are identical. Figure 1f plots a CDF of Jaccard index values for the PII leaked by each service's Web and app versions. We find that the types of PII leaked by Web- and app-based versions of the same service *share nothing in common more than half of the time.* Overall, 80-90% of services share only 50% of the PII types leaked across app and Web.

The previous result is perhaps expected because app and Web A&A systems have different PII available to them, and thus use different mechanisms for tracking. For example, app-based tracking can identify sessions belonging to the same user via a device's unique identifiers, while Web sites tend to use cookie IDs and cookie matching [10]. However, in many cases the differences in the types of PII leaks are substantial; for example Priceline leaked birthdays and gender from their Web site, but do not do so from either iOS or Android apps (each of which in turn leaks different PII).

In summary, we find that apps are more likely to leak more PII types than their Web counterparts, and most online ser-

[3] A "best practice" referred to as "encrypted at rest and in motion."



(a) A&A domains **(b)** A&A flows **(c)** A&A bytes

(d) Leak domains **(e)** Leaked identifiers **(f)** Jaccard of leaked identifiers

Figure 1: For subfigures (a)-(d), we find the differences between app and Web versions of the same service, in terms of A&A domains visited, number of flows to them, and the number of bytes they consume, and the number of domains they leak PII to. Subfigures (e)-(f) compare the set of identifiers leaked by app and Web versions of each service.

vices leak substantially different PII over the two media. We believe this occurs due to the fact that apps and Web sites often have different mechanisms for data collection, different analytics companies, and different development teams. Interestingly, the services we tested provide *the same functionality* over app and Web, and should in theory be able to provide (at a high level) uniform data collection policies across platforms. The fact that they do not provides an opportunity for users to make informed privacy decisions when choosing whether to install an app or use a Web site (independent of the reasons behind these differences).

Recipients of PII Leaks. To understand how pervasively user PII is exposed to other parties, we analyze our dataset according to which third party is contacted (via Web or app), and identify whether app- or Web-based trackers collect more or less of a certain type of PII. We focus on the top-20 A&A domains receiving PII (sorted by total leaks in our dataset). Table 2 shows each domain (absent its top-level domain), the number of services that contact it, the average number of leaks per service, and the number of leaked identifiers. We observe significant overlap between the apps and Web sites that contact each A&A domain, revealing that services tend to utilize the same trackers and ad networks across platforms.

Notably, the A&A domain receiving the most leaks (Amobee) is used by the fewest services (1). Further, the third column group shows that Amobee receives a similar set of PII over app and Web (intersection set size is two). In addition, we find that Facebook is the most pervasively contacted domain across our tested apps.

Interestingly, with few exceptions, top A&A domains collect at least one type of PII from apps that are not collected via Web sites. Thus, third-parties are leveraging different platforms to expand the set of data that they collect about users. We also see a small number of cases of platform-specific data collection, e.g., YieldMo only collects PII from apps in our set of services.

A&A Domain	# of Services: App	∩	Web	Avg. Leaks: App	Web	Leaked Identifiers: App	∩	Web
amobee	1	1	1	517.0	314.0	3	2	2
moatads	9	7	12	61.4	0.2	1	1	1
vrvm	2	0	0	136.0	0.0	3	0	0
google-analytics	35	32	41	1.8	2.7	1	1	2
facebook	38	36	41	3.7	0.4	2	0	1
groceryserver	1	1	1	154.0	0.0	1	0	0
serving-sys	10	4	6	15.3	0.0	1	0	0
googlesyndication	16	14	23	7.0	0.8	1	1	1
thebrighttag	4	2	4	29.5	0.0	2	0	0
tiqcdn	5	5	9	16.0	3.1	1	1	1
marinsm	1	1	3	96.0	1.0	1	0	1
criteo	7	6	22	8.9	1.1	2	1	2
2mdn	14	9	17	5.8	0.0	1	0	0
monetate	1	1	2	74.0	0.0	1	0	0
247realmedia	1	1	2	48.0	12.0	1	0	1
krxd	7	6	13	8.3	0.0	3	0	0
doubleverify	3	2	7	19.3	0.0	1	0	0
cloudinary	1	1	1	0.0	58.0	0	0	1
webtrends	1	1	1	56.0	0.0	1	0	0
liftoff	1	0	0	54.0	0.0	2	0	0

Table 2: Top-20 A&A domains, sorted by total leaks.

Last, we focus on how each type of PII is leaked across Web sites and apps in Table 3 (again, sorted by total leaks). We see that locations, names, and unique tracking IDs are most commonly leaked, with device-specific IDs being leaked only over apps. The first column group shows that the apps and Web sites leaking specific pieces of PII have relatively low overlap (except for location), reinforcing our finding that services may have very different privacy profiles across platforms. Similarly, the third column group shows that each type of PII is leaked to a significant number of domains by both apps and Web sites, though the domains *in common* between the two is a fraction of the total.

In summary, we find that there is no clear winner in terms of privacy-footprint between apps and their Web counterparts. Services leak significant information on both platforms, but typically not the same information.

PII	# of Services: App	∩	Web	Avg. Leaks: App	Web	Domains Leaked To: App	∩	Web
Location	30	21	26	367.7	295.2	84	37	76
Name	9	8	16	77.1	138.2	11	7	26
Unique ID	40	0	0	39.0	0.0	65	0	0
Username	3	1	5	23.0	89.8	4	2	10
Gender	4	1	8	2.8	25.0	4	1	11
Phone #	3	1	2	12.7	60.5	3	1	2
Email	11	3	8	2.2	15.5	10	2	8
Device Name	15	0	0	2.7	0.0	13	0	0
Password	4	2	3	2.8	1.7	4	2	2
Birthday	1	0	1	1.0	3.0	1	0	2

Table 3: PII, sorted by total leaks.

5. CONCLUDING DISCUSSION

This paper asks a simple question—are apps or Web sites better for privacy?—and finds the answer not at all straightforward. Several clear trends emerged: more domains are contacted from Web sites, and more device identifiers were leaked from apps. However, we also found a pervasive tracking ecosystem that exposes users' PII across both Web and app versions of the same service, and across different services. In short, there is no single answer to the seminal question in this work; rather, the answer depends on user preferences and priorities for controlling access to their PII. Our analysis provides the necessary data to inform custom recommendations for privacy via:

`https://recon.meddle.mobi/appvsweb/`

There are a number of interesting topics for future research. For example, we would like to understand cross-service PII leaks, as well as provide users with actionable information about how leaked PII can be used by other parties to build profiles about them. An interesting question is how effective are existing browser privacy protection tools in light of our findings, and how we might augment ReCon to provide improved protection in the mobile environment.

Acknowledgements

We thank the anonymous reviewers and our shepherd Theo Benson for their helpful feedback. This work was partially supported by the Data Transparency Lab, and by NSF grants IIS-1408345 and IIS-1553088. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

6. REFERENCES

[1] App Annie App Store Stats. `http://www.appannie.com/`.

[2] EasyList. `https://easylist.github.io/`.

[3] Mitmproxy. `https://mitmproxy.org/`.

[4] Android developer dashboard, April 2016. `http://developer.android.com/about/dashboards/index.html`.

[5] Acar, G., Eubank, C., Englehardt, S., Juarez, M., Narayanan, A., and Diaz, C. The web never forgets: Persistent tracking mechanisms in the wild. In *Proc. of CCS* (2014).

[6] Agarwal, Y., and Hall, M. ProtectMyPrivacy: Detecting and Mitigating Privacy Leaks on iOS Devices Using Crowdsourcing. In *Proc. of MobiSys* (2013).

[7] Arzt, S., Rasthofer, S., Fritz, C., Bodden, E., Bartel, A., Klein, J., Le Traon, Y., Octeau, D., and McDaniel, P. FlowDroid: Precise Context, Flow, Field, Object-sensitive and Lifecycle-aware Taint Analysis for Android Apps. In *Proc. of PLDI* (2014).

[8] Ayenson, M., Wambach, D. J., Soltani, A., Good, N., and Hoofnagle, C. J. Flash cookies and privacy ii: Now with html5 and etag respawning. *Available at SSRN 1898390* (2011).

[9] Azim, T., and Neamtiu, I. Targeted and Depth-first Exploration for Systematic Testing of Android Apps. In *Proc. of OOPSLA* (2013).

[10] Bashir, M. A., Arshad, S., Robertson, W., and Wilson, C. Tracing Information Flows Between Ad Exchanges Using Retargeted Ads. In *Proceedings of the 25th USENIX Security Symposium* (2016).

[11] Cahn, A., Alfeld, S., Barford, P., and Muthukrishnan, S. An empirical study of web cookies. In *Proc. of WWW* (2016).

[12] Chen, X., and Zhu, S. DroidJust: Automated Functionality-aware Privacy Leakage Analysis for Android Applications. In *Proc. of WiSec* (2015).

[13] Choudhary, S. R., Gorla, A., and Orso, A. Automated Test Input Generation for Android: Are We There Yet? In *Proc. of the IEEE/ACM International Conference on Automated Software Engineering (ASE)* (2015).

[14] Egele, M., Kruegel, C., Kirda, E., and Vigna, G. PiOS: Detecting Privacy Leaks in iOS Applications. In *Proc. of NDSS* (2011).

[15] Englehardt, S., Reisman, D., Eubank, C., Zimmerman, P., Mayer, J., Narayanan, A., and Felten, E. W. Cookies that give you away: The surveillance implications of web tracking. In *Proc. of WWW* (2015).

[16] Falahrastegar, M., Haddadi, H., Uhlig, S., and Mortier, R. The rise of panopticons: Examining region-specific third-party web tracking. In *Proc of. Traffic Monitoring and Analysis* (2014).

[17] Gibler, C., Crussell, J., Erickson, J., and Chen, H. AndroidLeaks: Automatically Detecting Potential Privacy Leaks in Android Applications on a Large Scale. In *Proc. of TRUST* (2012).

[18] Gill, P., Erramilli, V., Chaintreau, A., Krishnamurthy, B., Papagiannaki, K., and Rodriguez, P. Follow the money: Understanding economics of online aggregation and advertising. In *Proc. of IMC* (2013).

[19] Hao, S., Liu, B., Nath, S., Halfond, W. G., and Govindan, R. PUMA: Programmable UI-Automation for Large-Scale Dynamic Analysis of Mobile Apps. In *Proc. of MobiSys* (2014).

[20] Hao, S., Liu, B., Nath, S., Halfond, W. G., and Govindan, R. PUMA: Programmable UI-automation for Large-scale Dynamic Analysis of Mobile Apps. In *Proc. of MobiSys* (2014).

[21] Jeon, J., Micinski, K. K., and Foster, J. S. SymDroid: Symbolic Execution for Dalvik Bytecode. Tech. Rep. CS-TR-5022, University of Maryland, College Park, 2012.

[22] Kamkar, S. Evercookie - virtually irrevocable persistent cookies., September 2010. `http://samy.pl/evercookie/`.

[23] Kim, J., Yoon, Y., Yi, K., and Shin, J. SCANDAL: Static Analyzer for Detecting Privacy Leaks in Android Applications. In *Proc. of MoST* (2012).

[24] Kohno, T., Broido, A., and Claffy, K. Remote physical device fingerprinting. *IEEE Transactions on Dependable and Secure Computing 2*, 2 (2005), 93–108.

[25] Krishnamurthy, B., Malandrino, D., and Wills, C. E. Measuring privacy loss and the impact of privacy protection in web browsing.

[26] Krishnamurthy, B., Naryshkin, K., and Wills, C. Privacy diffusion on the web: A longitudinal perspective. In *Proc. of WWW* (2009).

[27] Krishnamurthy, B., and Wills, C. Privacy leakage vs. protection measures: the growing disconnect. In *Proc. of W2SP* (2011).

[28] Li, T.-C., Hang, H., Faloutsos, M., and Efstathopoulos, P. Trackadvisor: Taking back browsing privacy from third-party trackers. In *Proc. of PAM* (2015).

[29] Liu, Y., Song, H. H., Bermudez, I., Mislove, A., Baldi, M., and Tongaonkar, A. Identifying personal information in internet traffic. In *Proceedings of the 3rd ACM Conference on Online Social Networks (COSN'15)* (Palo Alto, CA, November 2015).

[30] Lu, L., Li, Z., Wu, Z., Lee, W., and Jiang, G. CHEX: Statically Vetting Android Apps for Component Hijacking Vulnerabilities. In *Proc. of ACM CCS* (2012).

[31] Machiry, A., Tahiliani, R., and Naik, M. Dynodroid: An Input Generation System for Android Apps. In *Proc. of the Joint Meeting on Foundations of Software Engineering (ESEC/FSE)* (2013).

[32] McDonald, A. M., and Cranor, L. F. Americans' attitudes about internet behavioral advertising practices. In *Proc. of WPES* (2010).

[33] Mowery, K., and Shacham, H. Pixel perfect: Fingerprinting canvas in html5. In *Proc. of W2SP* (2012).

[34] Nikiforakis, N., Kapravelos, A., Joosen, W., Kruegel, C., Piessens, F., and Vigna, G. Cookieless monster: Exploring the ecosystem of web-based device fingerprinting. In *Proc. of IEEE Symposium on Security and Privacy* (2013).

[35] Olejnik, L., Castelluccia, C., and Janc, A. Why Johnny Can't Browse in Peace: On the Uniqueness of Web Browsing History Patterns. In *Proc. of HotPETs* (2012).

[36] Papaodyssefs, F., Iordanou, C., Blackburn, J., Laoutaris, N., and Papagiannaki, K. Web identity translator: Behavioral advertising and identity privacy with wit. In *Proc. of HotNets* (2015).

[37] Rao, A., Kakhki, A. M., Razaghpanah, A., Li, A., nad Arnaud Legout, D. C., Mislove, A., and Gill, P. Meddle: Enabling Transparency and Control for Mobile Internet Traffic. *JoTS*, 2015103003 (October 2015).

[38] Ren, J., Rao, A., Lindorfer, M., Legout, A., and Choffnes, D. R. ReCon: Revealing and controlling privacy leaks in mobile network traffic. In *Proc. of MobiSys* (2016).

[39] Roesner, F., Kohno, T., and Wetherall, D. Detecting and defending against third-party tracking on the web. In *Proc. of NSDI* (2012).

[40] The Wall Street Journal. What They Know - Mobile. `http://blogs.wsj.com/wtk-mobile/`, December 2010.

[41] Turow, J., Hennessy, M., and Draper, N. The tradeoff fallacy: How marketers are misrepresenting american consumers and opening them up to exploitation. Report from the Annenberg School for Communication, June 2015. `https://www.asc.upenn.edu/sites/default/files/TradeoffFallacy_1.pdf`.

[42] Vallina-Rodriguez, N., Shah, J., Finamore, A., Grunenberger, Y., Papagiannaki, K., Haddadi, H., and Crowcroft, J. Breaking for commercials: Characterizing mobile advertising. In *Proc. of IMC* (2012).

[43] Xia, M., Gong, L., Lyu, Y., Qi, Z., and Liu, X. Effective Real-time Android Application Auditing. In *IEEE Symposium on Security and Privacy* (2015).

[44] Yan, L. K., and Yin, H. DroidScope: Seamlessly Reconstructing the OS and Dalvik Semantic Views for Dynamic Android Malware Analysis. In *Proc. of USENIX Security* (2012).

[45] Yang, Z., Yang, M., Zhang, Y., Gu, G., Ning, P., and Wang, X. S. AppIntent: Analyzing Sensitive Data Transmission in Android for Privacy Leakage Detection. In *Proc. of ACM CCS* (2013).

[46] Zhang, Y., Yang, M., Xu, B., Yang, Z., Gu, G., Ning, P., Wang, X. S., and Zang, B. Vetting undesirable behaviors in Android apps with permission use analysis. In *Proc. of ACM CCS* (2013).