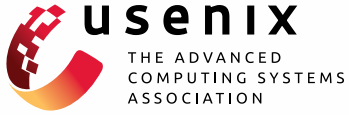

Anatomization and Protection of Mobile Apps' Location Privacy Threats

Kassem Fawaz, Huan Feng, and Kang G. Shin, ***University of Michigan***

https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/fawaz

This paper is included in the Proceedings of the 24th USENIX Security Symposium

August 12–14, 2015 • Washington, D.C.

ISBN 978-1-931971-232

Open access to the Proceedings of the 24th USENIX Security Symposium is sponsored by USENIX

Anatomization and Protection of Mobile Apps' Location Privacy Threats

Kassem Fawaz, Huan Feng, and Kang G. Shin
The University of Michigan
{*kmfawaz, huanfeng, kgshin*}*@umich.edu*

Abstract

Mobile users are becoming increasingly aware of the privacy threats resulting from apps' access of their location. Few of the solutions proposed thus far to mitigate these threats have been deployed as they require either app or platform modifications. Mobile operating systems (OSes) also provide users with location access controls. In this paper, we analyze the efficacy of these controls in combating the location-privacy threats. For this analysis, we conducted the first location measurement campaign of its kind, analyzing more than 1000 free apps from Google Play and collecting detailed usage of location by more than 400 location-aware apps and 70 Advertisement and Analytics (A&A) libraries from more than 100 participants over a period ranging from 1 week to 1 year. Surprisingly, 70% of the apps and the A&A libraries pose considerable profiling threats even when they sporadically access the user's location. Existing OS controls are found ineffective and inefficient in mitigating these threats, thus calling for a finer-grained location access control. To meet this need, we propose `LP-Doctor`, a light-weight user-level tool that allows Android users to effectively utilize the OS's location access controls while maintaining the required app's functionality as our user-study (with 227 participants) shows.

1 Introduction

Mobile users are increasingly aware of the privacy threats caused by apps' access of their location [12, 42]. According to recent studies [14, 17, 42], users are also taking measures against these threats ranging from changing the way they run apps to disabling location services all together on their mobile devices. How to mitigate location-privacy threats has also been researched for some time. Researchers have proposed and even implemented location-privacy protection mechanisms (LPPMs) for mobile devices [2, 6, 12, 20, 30]. However, few of them have been deployed as they require app or system-level modifications, both of which are unappealing/unrealistic to the ordinary users.

Faced with location-privacy threats, users are left only with whatever controls the apps and OSes provide. Some, but not all, apps allow the users to control their location access. OSes have been improving on this front. iOS includes a new permission to authorize location access in the background, or when the app is not actively used. Also, iOS, Windows OS, and Blackberry (Android to follow suit) utilize per-app location-access permissions. The user authorizes location access at the very first time an app accesses his location and has the option to change this decision for every subsequent app invocation. We want to answer two important questions related to this: *(i) are these controls effective in protecting the user's location privacy and (ii) if not, how can they be improved at the user level without modifying any app or the underlying OS?*

To answer these questions, we must understand the location-privacy threats posed by mobile apps. This consists of understanding the apps' location-access patterns and their usage patterns. For this, we instrumented and analyzed the top 1165 downloaded free apps (that require location-access permissions) from Google Play to study their location-access patterns. We also studied the behavior of Advertisement and Analytics (A&A) libraries, such as Flurry, embedded in the apps that might access location. We analyzed only those apps/libraries that access location through Android's official location APIs. While some apps/libraries might circumvent the OS in accessing location, it is an orthogonal problem to that addressed in this paper.

We then analyzed the users' app-usage patterns by utilizing three independent datasets. First, we collected and analyzed app-tagged location traces through a 10-month data collection campaign (Jan. 2013—Nov. 2013) for 24 Android smartphone users. Second, we recruited 95 Android users through PhoneLab [31], a smartphone mea-

surement testbed at New York State University at Buffalo, for 4 months. Finally, we utilized the dataset from Livelab at Rice University [34] that contains app-usage and location traces for 34 iPhone users for over a year.

Utimately, we were able to evaluate the privacy threats posed by 425 apps and 77 third-party libraries. 70% of the apps are found to have the potential of posing profiling threats that have not yet been adequately studied or addressed before [15, 16, 25, 41]. Moreover, the A&A libraries pose significant profiling threats on more than 80% of the users as they aggregate location information from multiple apps. Most of the users are unaware of these threats as they can't keep track of exposure of their location information. The issue becomes more problematic in the case of A&A libraries where users are oblivious to which apps these libraries are packed in and whether they are receiving location updates.

Given the nature of the threats, we studied the effectiveness of the existing OS controls. We found that these controls are capable of thwarting only a fraction of the underlying privacy threats, especially tracking threats. As for profiling, the user only has the options of either blocking or allowing location access. These two options come at either of the two extremes of the privacy–utility spectrum: the user either enjoys full privacy with no utility, or full utility with no privacy. As for A&A libraries, location accesses from a majority of the apps must be blocked to thwart the location-privacy threats caused by these libraries.

The main problem arises from the user's inability to exercise fine-grained control on when an app should receive a location update. The interface provided by existing controls makes it hard for the user to enforce location-access control on a per visited place/session basis. Even if the user can dynamically change the control of location access, he cannot estimate the privacy threats at runtime. The location-privacy threat is a function of the current location along with previously released locations. This makes it difficult to estimate the threat for apps and even harder for A&A libraries.

To fill this gap, we propose `LP-Doctor`, a user-level app, to protect the location privacy of smartphone users, which offers three salient features. First, `LP-Doctor` evaluates the privacy threat that the app might pose before launching it. If launching the app from the current location poses a threat, then it acts to protect the user's privacy. It also warns the user of the potential threat in a non-intrusive manner. Second, `LP-Doctor` is a *user-level* app and does not require any modification to the underlying OS or other apps. It acts as a control knob for the underlying OS tools. Third, `LP-Doctor` lets the user control, for each app, the privacy–utility tradeoff by adjusting the protection level while running the app.

We implemented `LP-Doctor` as an Android app that can be downloaded from Google Play. The privacy protection that `LP-Doctor` provides comes at a minimal performance overhead. We recruited 227 participants through Amazon Mechanical Turk and asked them to download and use `LP-Doctor` from Google Play. The overwhelming majority of the participants reported little effect on the quality of service and user experience. More than 77% of the participants indicated that they would install `LP-Doctor` to protect their location privacy.

In summary, we make the following main contributions:

- The first location data collection campaign of its kind to measure, analyze, and model location-privacy threats from the apps' perspectives (Sections 3–6);
- Evaluation of the effectiveness of OS's location privacy controls by anatomizing the location-privacy threats posed by the apps (Sections 7–8);
- Design, implementation and evaluation of a novel user-level app, `LP-Doctor`, based on our analysis to fill the gaps in existing controls and improve their effectiveness (Section 9).

2 Related Work

App-Based Studies: To the best of our knowledge, this is the first attempt to quantify and model location privacy from the apps' perspective. Researchers already concluded that many mobile apps and A&A libraries leak location information about the users to the cloud [5, 23, 38]. These efforts are complimentary to ours; we study the quantity and quality of location information that the apps and libraries locally gather while assuming that they may leak this information outside the device.

Analysis of Location Privacy: Influenced by existing location datasets (vehicular traces, cellular traces, etc.), most of the existing studies view location privacy in smartphones as if there were only one app *continuously* accessing a user's location [7, 11, 25, 26, 29, 33, 41]. Researchers also proposed mechanisms [28, 29, 32] (their effectiveness analyzed by Shokri *et al.* [36]) to protect against the resulting tracking-privacy threats. Such mechanisms have shown to be ineffective in thwarting the profiling threats [41] which are more prevalent as we will show later.

Researchers started considering sporadic location-access patterns as a source of location-privacy threat that calls for a different treatment than the continuous case [4]. Still, existing studies focus mostly on the tracking threat [3, 35]. The only exception to this is the work by Freudiger *et al.* [15]. They assessed the erosion of the

user's privacy from sporadic location accesses as the portion of the PoIs identified after downsampling the continuous location trace. In this paper, we propose a formal metric to model the profiling threats. Also, we show that an app's location-access behavior can't be modeled as simply downsampling the user's mobility.

Location-Privacy Protection Proposals: Several solutions have been proposed to protect mobile users' location privacy. MockDroid [6] allows for blocking apps' location access to protect the user's location privacy. LP-Guardian [12] is another system aiming at protecting the user's location privacy by incorporating a myriad of mechanisms. Both systems require platform modifications, hindering their deployment. Other mechanisms, such as Caché [2] and the work by Micinski *et al.* [30], provide apps with coarsened locations but require modifications to the apps. Koi [20] proposed a location privacy enhancing system that utilizes a cloud service, but requires developers to use a different API to access location. Apps on Google Play such as PlaceMask and Fake GPS Location Spoofer rely on the user to manually feed apps with fake locations, which reduce their usability.

Finally, researchers have proposed improved permission models for Android [1, 24]. In their models, the users are aware of how much the apps access their location and have the choice to enable/disable location access for each app (AppOps provided such functionality in Android 4.3). `LP-Doctor` improves on these in three ways. First, it provides a privacy model that maps each app's location access to a privacy metric. This model includes more information than just the number of location accesses by the app. Second, `LP-Doctor` makes some decisions on behalf of the users to avoid interrupting their tasks and to make privacy protection more usable. Third, `LP-Doctor` employs *per-session* location-access granularity which achieves a better privacy–utility tradeoff.

3 Background and Data Collection

To study the efficacy of location-access controls of different mobile OSes, we had to first analyze location-privacy threats from the apps' perspectives. This includes studying how different apps collect the user's location. We conduct a data collection campaign to achieve this using the Android platform. Our results, however, can be generalized to other mobile platforms like iOS.

3.1 Location-Access Controls

Each mobile platform provides users with a set of location-access controls to mitigate possible location-privacy threats. Android (prior to Android M) provides a one-time permission model that allows users to authorize location access. Once the user approves the permission

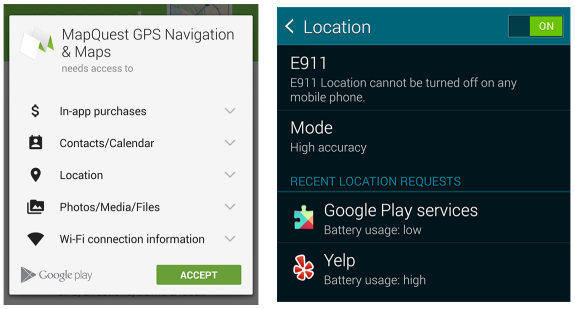

Figure 1: Android's permission list (left) and location settings (right).

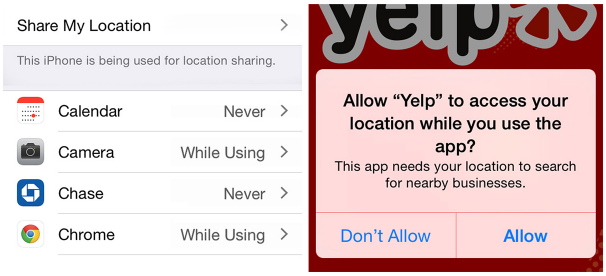

Figure 2: iOS's location settings (left) and prompts (right).

list (Fig. 1–left) for the app, it is installed and the permissions can't be revoked. It also provides a global location knob (Fig. 1–right) to control location services. The user can't exercise per-app location-access control.

Other platforms, such as Blackberry OS and iOS, provide finer-grained location permissions. Each app has a settings menu (Fig. 2–left) that indicates the resources it is allowed to access, including location. The user can at any point of time revoke resource access by any app. The first time an app accesses location, the OS prompts the user to authorize location access for the app in the current and future sessions (Fig. 2–right). Also, Google, starting from Android M, will provide a similar permission model (an evolution of the previously deployed AppOps in Android 4.3) to control access of location and other resources. At present, iOS provides the users with an additional option to authorize location access in the background to prevent apps from tracking users.

In the rest of this paper, we study the following controls: (1) one-time location permissions, (2) authorization of location access in the background, and (3) finer-grained per-app permissions.

3.2 System Model

We study location-privacy threats through apps and A&A libraries that access the user's location. These apps and libraries then provide the service, and keep the location

records indexed by a user ID, such as MAC address, Android ID, IMEI, etc.

We assume that the app/library communicates all of the user's location samples to the service provider.[1] This allows us to model the location-privacy threats caused by apps/libraries in the worst-case scenario. The app is the only means by which the service provider can collect the user's location updates. We don't consider cases where the service provider obtains the user's location via side channels other than the official API, e.g., an app reads the nearby access points and sends them to a localization service, such as skyhook.

We preclude system and browsing apps from our study for the following reasons. System apps are part of the OS that already has access to the user's location all the time. Hence, analyzing their privacy implications isn't very informative. As for the browsing apps, the location sink might be a visited website as well as the browser itself. We decided not to monitor the user's web history during the data collection for privacy concerns. Also, app-usage patterns differ from browsing patterns. The conclusions derived for the former don't necessarily translate to those for the latter.

3.3 App and A&A libraries Analysis

In February 2014, we downloaded the top 100 apps of each of Google Play's 27 app categories. We were left with 2588 unique apps, of which 1165 apps request location permissions. We then instrumented Android to intercept every location access invoked by both the app and the packed A&A libraries.

The main goal of this analysis was to unravel the situations in which an app accesses location and whether it feeds a packed A&A library. In Android, the app could be running in the foreground, cached in the background, or as a service. Using a real device, we ran every app in foreground, moved it to background, and checked if it forked a service, while recording its location requests.

Apps running in the foreground can access location spontaneously or in response to some UI event. So, we ran every app in two modes. In the first mode, the app runs for a predefined period of time and then closes, while in the second, we manually interact with each app to trigger the location-based functionality. Finally, we analyzed the functionality of every app and the required location granularity to achieve this functionality.

3.4 Data Collection

As will be evident in Section 4, the app-usage pattern is instrumental in determining the underlying location-privacy threats. We collected the app-usage data using an app that we developed and published on Google Play. Our study was deemed as not-requiring an IRB oversight by the IRB at our institution; all the data we collected is anonymous. Also, we clustered the participants' location on the device to extract their visited places. We define the "place" as a center location with a radius of 50m and a minimum visit time of 5 min. Then, we logged place IDs instead of absolute location samples to further protect the participants' privacy.

PhoneLab: PhoneLab [31] is a testbed, deployed at the NY State University at Buffalo, composed of 288 smartphone users. PhoneLab aims to free the researchers from recruiting participants by providing a diverse set of participants, which leads to stronger conclusions.

We recruited 95 participants to download and run our app for the period between February 2014 and June 2014. We collected detailed usage information for 625 apps, of which 218 had location permissions and were also part of the apps inspected in the app-analysis stage.

Our Institution: The second set consists of 24 participants whom we recruited through personal relations and class announcements. We launched this study from January 2013 till November 2013, with the participation period per user varying between 1 week and 10 months. From this set, we collected usage data of 256 location-aware apps.

We also collected location access patterns of some apps from a subset of the participants. We handed 11 participants Galaxy Nexus devices with an instrumented Android (4.1.2) that recorded app-tagged location accesses. We measured how frequently do ordinary users invoke location-based functionality of apps that don't spontaneously access location (e.g., Whatsapp).

LiveLab: Finally, we utilize the Livelab dataset [34] from Rice University. This dataset contains the app usage and mobility records for 34 iPhone users over the course of a year (2010). We post-processed this dataset to map app-usage records to the location where the apps were invoked. We only considered those apps that overlapped with our Android dataset (35 apps).

4 Location-Access Patterns

We address the location-access patterns by analyzing how different apps collect location information while running in foreground and background. The former represents the state where the user actively interacts with the app, while the latter represents the case where the app runs in the background either as cached by Android or as a persistent service.

As evident from Table 1, 74% of the apps solely access location when running in the foreground, while only 3% continuously access the user's location in the back-

[1] We refer to both the app developers and A&A agencies as the service provider.

Table 1: Location-access patterns for smartphone apps according to Android location permissions

	Fore. (%)	Cached (%)	Back. (%)	None (%)	Gran. Coarse (%)
Coarse	71	6	1	22	100
Fine	74	14	4	12	48
All	74	12	3	14	66

Table 2: Location-access patterns for A&A libraries

Total	No Location Access	App Feeds Location	Auto Location Access		
			Coarse	Fine	Both
77	22	17	3	2	33

ground. Around 70% of the apps accessing location in the foreground spontaneously perform such access preceding any user interaction. Examples of these apps include Angry Birds, Yelp, Airbnb, etc.

Android caches the app when the user exits it; depending on the app's behavior it might still access location; only 12% of the apps access the user's location when they are cached. Interestingly, for 14% of the apps, we didn't find any evidence that they access location in any state.

We also analyzed the location-based functionality of every app and the required location granularity to achieve such functionality. We focused on two location granularity levels: *fine* and *coarse*. A fine location sample is one with block-level granularity or higher, while coarse location is that with zipcode-level granularity or lower. We manually interacted with each app to assess the change in its functionality while feeding it locations with different granularity. We show the percentage of the apps that can accommodate coarse location without noticeable loss of app functionality in Table 1 under the column titled *Gran. Coarse*. One can notice that apps abuse the location permissions: 48% of the apps requesting fine location permissions can accommodate locations with coarser granularity without loss of functionality.

Finally, we analyzed the packed A&A libraries in these apps. We were able to identify 77 of such libraries packed in these apps. Table 2 shows basic statistics about these libraries. Most (more than 70%) libraries require location access where some are fed location from the apps (22%). The rest of the libraries automatically access location where 3 of them require coarse location permissions, 2 require fine permissions, and the rest don't specify a permission. Also, these libraries are included within more than one location-aware app giving them the ability to track the user's location beyond what a single app can do. For example, of 1165 analyzed apps, Google Ads is packed within 499 apps, Flurry within 325 apps, Medialets within 35 apps, etc.

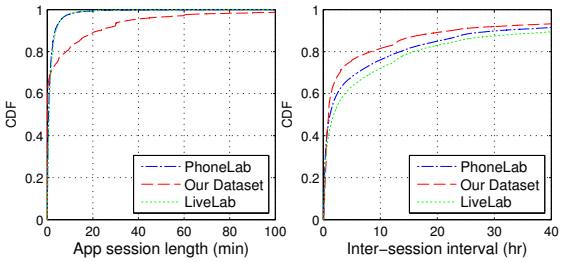

Figure 3: The distribution of app session lengths (left) and inter-session intervals (right) for the three datasets.

5 App-Usage Patterns

As apps mostly access users' location in the foreground, the app-usage patterns (the way that users invoke different apps) help determine how much location information each app collects. Apps are shown to sporadically sample the user's location based on two facts. First, an app session is equivalent to the place visited during the session. Second, apps' inter-session intervals follow a Pareto-law distribution.

For foreground apps, we define a session as a single app invocation—the period of time in which a user runs the app then exits it. The session lengths are not long enough to cover more than one place the user visits, where 80% of these app sessions are shorter than 10 minutes (the left plot of Fig. 3). We confirmed this from our PhoneLab dataset; 98% of the app sessions started and ended at the same place.

This allows for collapsing an app session into one location-access event. It doesn't matter what frequency the app polls the user's location with. As long as the app requests the user's location at least once, while it is running in the foreground, it will infer that the user visited that location. We thus ignore the location-access frequency of foreground apps, and instead focus on the app-usage patterns.

We define the inter-session time as the interval separating different invocations (sessions) of the same app by the same user. The right plot of Fig. 3 shows the distribution of the inter-session intervals for the three datasets. More than 50% of the app sessions were separated by at least one hour.

We also found that the inter-session intervals follow a Pareto-law distribution rather than a uniform distribution. This indicates that apps don't sample the user's location uniformly, indicating that existing models for apps' location access don't match their actual behavior.

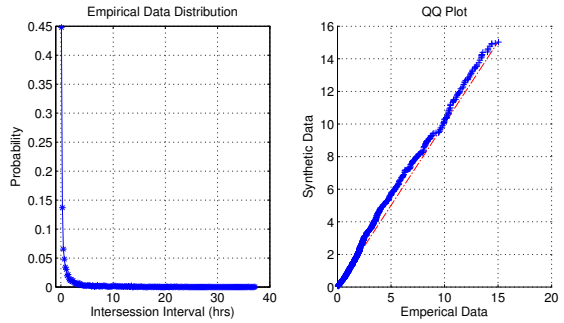

Figure 4: The distribution of the inter-session times for Facebook in Livelab dataset (left), and the QQ plot of this distribution versus a Pareto law distribution (right).

Fig. 4 shows the distribution of the inter-session intervals of a user running Facebook. It is evident that the distribution of the inter-session intervals decays linearly with respect to the increase of inter-session intervals. We observed a similar trend with all other apps. This suggests that the data decays according to a Pareto law (QQ plot in Fig. 4). We followed the guidelines outlined by Clauset *et al.* [10] to fit the data to the truncated Pareto distribution. Three parameters (L, H, and α) define the truncated Pareto law distribution:

$$p_X(x) \begin{cases} \frac{(-\alpha-1)L^{-\alpha-1}x^{\alpha}}{1-\left(\frac{L}{H}\right)^{-\alpha-1}} & \text{if } L \leq x \leq H \\ 0 & \text{otherwise.} \end{cases}$$

After fitting the data, more than 97% of the app-usage models are found to have α between -1 and -1.5. According to Vagna *et al.* [40], Pareto law fits different human activity models with α between -1 and -2.

6 Privacy Model

Here we model the privacy threats caused by mobile apps'/libraries' access of the user's location.

6.1 Preliminaries

Below we describe the models of user mobility, app-usage, and adversaries that we will use throughout the paper.

User Mobility Model: We assume there is a region (e.g., city) of interest which includes set of places that the user can visit. So, a domain of interest is represented by the set Pl of all the places available in that domain: $Pl = \{pl_1, pl_2, pl_3 \ldots\}$. Under this model, the user visits a set of places, $U_{Pl} \subseteq Pl$, as part of his daily life, spends time at pl_i running some apps and then moves to another place pl_j. We alternatively refer to these places as the user's *Points of Interest* (PoIs).

We associate every place pl_i with a visit probability of p_i, reflecting the portion of time the user spends at pl_i. The user's *mobility profiles* are defined as the set, U_{pl}, of places he visited and the probability, p_i, of visit to each place. The mobility profile is unique to each user since a different user visits a different set of places with a different probability distribution [41].

App-Usage Model: In Section 5, we showed that each app session is equivalent to an observation of the user's visit to a place. The app accumulates observations of the set of places that the user visits. The app will eventually observe that a user visited a certain place pl_i for c_{pl_i} times. So, we view the app as a random process that samples the user's entire location trace and outputs a histogram of places of dimension $|U_{Pl}|$. Each bin in the histogram is the number of times, c_{pl_i}, the app observes the user at that specific place. The total number of visits is represented as $N = \sum_{i=1}^{|U_{Pl}|} c_{pl_i}$.

The histogram represents the app's view of the user's mobility. Most apps don't continuously monitor user's mobility as they don't access location in the background. As such, they can't track users; the most these apps can get from a user is the histogram of the places he visited, which constitutes the source of location-privacy threats in this case.

Adversary Model: The adversary in our model is not necessarily a malicious entity seeking to steal the user's private information. It is rather a curious entity with possession of the user's location trace. The adversary will process and analyze these traces to infer more information about the user that allows for a more personalized service. This is referred to as *authentic apps* [39]. The objective of our analysis is to study the effect of the ordinary apps collecting location on the user's privacy.

Apps accessing location in the foreground can't track the user (Section 8). So, the adversary seeks to profile the user based on locations he visited. We use the term *profiling* to represent the process of inferring more information about the user through the collected location data. The profiling can take place at multiple levels, ranging from identifying preferences all the way to revealing the user's identity. Instead of modeling the adversary's profiling methods/attacks, we quantify the amount of information that location data provides the adversary with. The intuition behind our analysis of the profiling threat is that the more descriptive the app's histogram of the actual user's mobility pattern, the higher the threat is.

6.2 Privacy Metrics

Table 3 summarizes the set of metrics that we utilize to quantify the privacy threats that each app poses from its location access. The simplest metric is the fraction of the users' PoIs the app can identify [15]. We evaluate this

Table 3: The metrics used for evaluating the location privacy threats.

Metric	Description
PoI_{total}	Fraction of the user's PoIs
PoI_{part}	Fraction of the user's infrequently visited PoIs
$Prof_{cont}$	Distance between the user's histogram and mobility profile
$Prof_{bin}$	χ^2 test of the user's histogram fitting the mobility profile

metric by looking at the apps' actual collected location traces, rather than a downsampled location trace. We will henceforth refer to this metric as PoI_{total}.

We also consider a variant of the metric (referred to as PoI_{part}) as the portion of the sensitive PoIs that the apps might identify. We define the sensitive PoIs as those that have a very low probability of being visited. These PoIs will exhibit abnormalities in the user's behavior. Research results in psychology [19, 21] indicated that people regard deviant (abnormal) behavior as being more private and sensitive. Places that an individual might visit that are not part of his regular patterns might leak a lot of information and are thus more sensitive in nature.

The histogram, as we mentioned before, is a sample of the user's mobility pattern. The second aspect of the profiling is quantifying how descriptive of the user's mobility pattern (original distribution) the app's histogram (sample) is.

For the purpose of our analysis and the privacy-preserving tool we propose later, we need two types of metrics. The first is a *continuous* metric, $Prof_{cont}$, that quantifies the profiling threat as the distance between the original distribution (mobility profile) and the sample (app's histogram). The second is a *binary* metric, $Prof_{bin}$, that indicates whether a threat exists or not.

For $Prof_{cont}$, we use the KL-divergence [27] as a measure of the difference (in bits) between the histogram (H) and the user's mobility pattern. The K-L divergence is given by $D_{\mathrm{KL}}(H\|p) = \sum_{i=1}^{|U_{Pl}|} H(i) \ln \frac{H(i)}{p_i}$, where $H(i)$ is the probability of the user visiting place pl_i based on the histogram, while p_i is the probability of the user visiting that place based on his mobility profile. The lower (higher) the value of $Prof_{cont}$, the higher (lower) the threat will be since the distance between the histogram and mobility pattern will be smaller (larger).

$Prof_{cont}$ is not useful in identifying histograms that pose privacy threats. There is no intuitive way by which a threshold can separate values that pose threats and those not posing any threat. So, we need a criterion indicating whether or not a threat exists based on the app's histogram. We use Pearson's Chi-square goodness of fit test to meet this need. This test indicates if the observed sample differs from the original (theoretical) distribution. Specifically, it checks if the null hypothesis of the sample originating from an original distribution can be accepted or not.

The test statistic, in our context, is $\chi^2 = \sum_{i=1}^{|U_{Pl}|} \frac{(c_{pl_i} - E_i)^2}{E_i}$ where $E_i = N.p_i$ is the expected number of visits to the place pl_i. The statistic converges to a Chi-squared distribution with $|U_{Pl}| - 1$ degrees of freedom when the null hypothesis holds. The test yields a p-value which if smaller than the significance level (α) then the null hypothesis can be rejected ($Prof_{bin} = 0$—no threat), else $Prof_{bin} = 1$, where null hypothesis can't be rejected, indicating the existence of a threat. In Sections 7 and 8, we employ the widely-used value of 0.05 as the significance level.

A&A libraries: can aggregate location information from the different apps in which they are packed and allowed to access location. We can thus view the histogram pertaining to an A&A library as the aggregate of the histograms of the apps in which the library is packed. We evaluate the same metrics for the aggregated histogram.

For the case of PoI_{total} and PoI_{part} metric, the aggregate histogram will be representative of the threat posed by the libraries. As for $Prof_{cont}$ and $Prof_{bin}$, we consider the aggregate histogram as well as the individual apps' histograms. The threat per library is the highest of that of the aggregate and individual histograms. The privacy threat posed by the library is at least as bad as that of any app that packs it in.

7 Anatomy

We now present the major findings from our measurement campaign. We analyze the location trace of each app and user, and hence, every data point in the subsequent plots belongs to an app–user combination. We constructed each app's histogram by overlaying its location-access pattern on its usage data for every user.

Privacy Threat Distribution: Fig. 5 shows the distributions of PoI_{total}, PoI_{part}, and $Prof_{cont}$ for both the apps and A&A libraries. As to PoI_{total}, most of the apps can identify at least 10% of the user's PoIs; while for 20% of the app–user combinations, apps were able to identify most of the user's PoIs. Apps can't identify all of the user's PoIs for two reasons: (1) not all apps access the user's location every time, as highlighted in Section 4, and (2) users don't run their apps from every place they visit. On the other hand, A&A libraries can identify more of the user's PoIs, with most of the libraries identifying at least 20% of the user's PoIs. Moreover, as the middle plots of Fig. 5 indicate, around 30% of the apps were able to identify some of the user's sensitive (less frequently visited) PoIs. More importantly, A&A libraries were able to identify more of the user's sensitive PoIs,

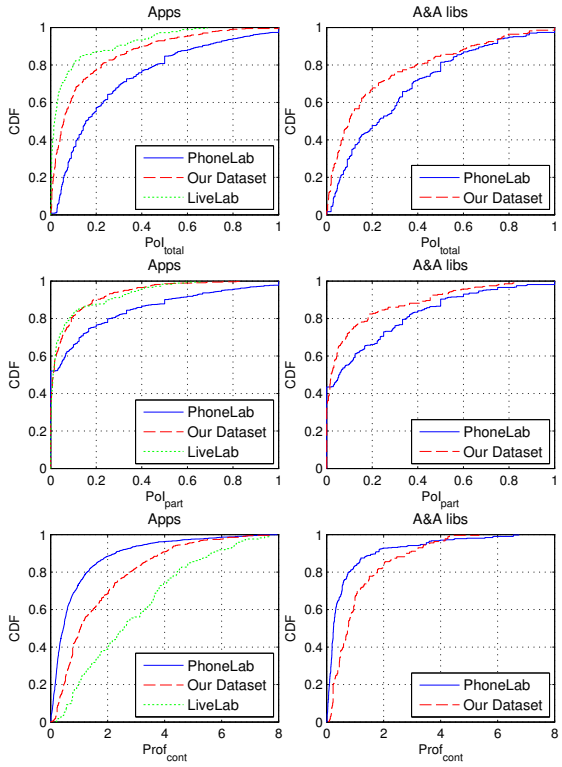

Figure 5: The distributions of PoI_{total} (top), PoI_{part} (middle), and $Prof_{cont}$ (bottom) for the apps (left) and A&A libraries (right) from our datasets.

indicating the level of privacy threats they pose.

The two bottom plots of Fig. 5 show the distributions of the profiling metric $Prof_{cont}$ for the foreground apps in the three datasets. The lower the value of the metric, the higher the privacy threat is. There are two takeaways from these two plots. First, apps do pose significant privacy threats; the distance between the apps' histogram and the user's mobility pattern is less than 1 bit in 40% of the app–user combinations for the three datasets. The second observation has to do with the threat posed by A&A libraries. It is clear from the comparison of the left and right plots that these libraries pose considerably higher threats. In more than 80% of user–library combinations, the distance between the observed histograms and the user's mobility profile is less than 1 bit.

Apps tend to even pose higher identification threats. As evident from Fig. 5, some apps can identify a relatively minor portion of the user's mobility which might not be sufficient to fully profile the user. Nevertheless, the portion of PoIs tend to be those users frequently visit (e.g., home and work) which may suffice to identify them [18, 25, 41]. This might not be a serious issue for those apps, such as Facebook, that can learn the user's home and work from other methods. Other apps

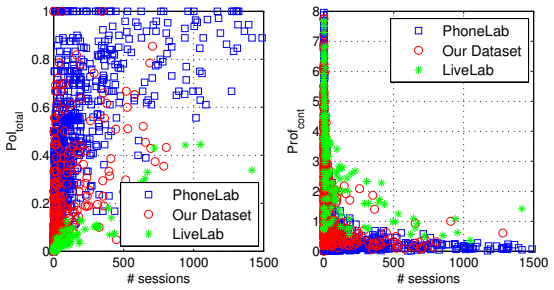

Figure 6: The distribution of PoI_{total} (left) and $Prof_{cont}$ (right) vs. the number of app sessions.

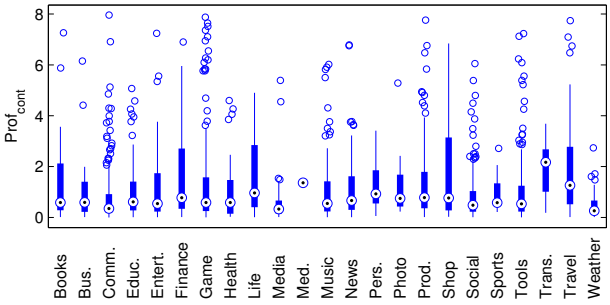

Figure 7: The distribution of $Prof_{cont}$ vs. app categories.

and libraries (e.g., Angry Birds), however, might infer the user's identity even when he anonymously uses them (without providing an identity or login information).

Fig. 5 also confirms our intuition in studying the location traces from the apps' perspective. If apps were to uniformly sample the user's mobility as has been assumed in literature, $Prof_{cont}$ should be mostly close to 0 (indicating no difference between the histogram and the mobility pattern), which is not the case.

Privacy Threats and App-Usage: We also evaluated the posed privacy threats vs. the app-usage rate as shown in Fig. 6. As evident from the plots, there is little correlation between the amount of posed threats and the app-usage rate. Apps that are used more frequently, do not necessarily pose higher threats, as user mobility, the app's location-access pattern, and the user's app-usage pattern affect the privacy threat.

With lower usage rates, both PoI_{total} and $Prof_{cont}$ vary significantly. Users with little diversity in their mobility pattern are likely to visit the same places more frequently. Even the same user could invoke apps differently; he uses some apps mostly at unfamiliar places (navigation apps), while using other apps more ubiquitously (gaming apps), thus enabling the apps to identify more of his PoIs.

Finally, we studied the distribution of the threat in relation to app categories. Fig. 7 shows that the threat level is fairly distributed across different app categories and

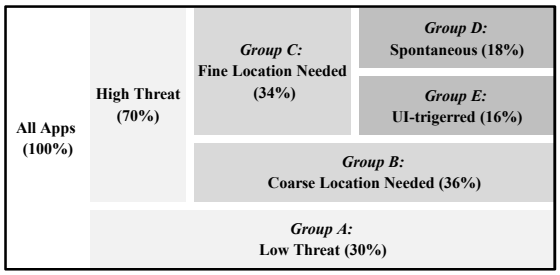

Figure 8: App categorization according to threat levels, location requirements, and location-access patterns.

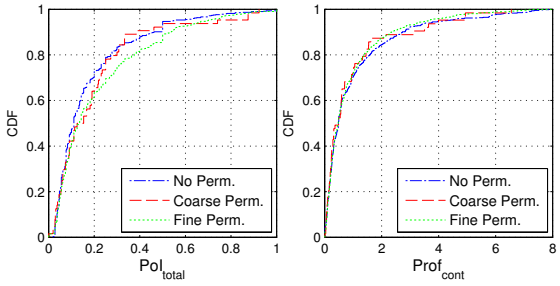

Figure 9: The distribution of PoI_{total} (left) and $Prof_{cont}$ (right) for PhoneLab apps with different permissions.

the same category. This confirms, again, that privacy threats result from multiple sources and are a function of both apps and users. Some app categories, however, pose lower threats on average. For example, transportation apps (including navigation apps) pose lower threats as users tend to use from unfamiliar places.

Threat Layout: Given the three datasets, we were able to analyze the profiling threats as posed by 425 location-aware apps (Fig. 8). For this part, we use $Prof_{bin}$ metric to decide which apps pose privacy threats and those which don't. As apps pose different threats depending on the users, we counted an app as posing a threat if it poses a privacy threat to at least one user. Only a minority of the apps (30%) pose negligible threats.

The rest of the apps pose a varying degree of profiling threat. We analyzed their functionality: 52% of such apps don't require location with high granularity to provide location-based functionality. For these apps, a zipcode- or city-level granularity would be more than enough (weather apps, games). This leaves us with 34% of the apps that require block-level or higher location granularity to provide usable functionality. These apps either spontaneously access location (18%) or in response to a UI event (16%).

8 OS Controls

Having presented an anatomy for the location-privacy threats posed by mobile apps, we are now ready to evaluate the effectiveness of existing OSes' location access controls in thwarting these threats.

Global Location Permissions: Android's location permissions attempt to serve two purposes: notification and control. They notify the user that the app he is about to install can access his location. Also, permissions aim to control the granularity by which apps access location. Apps with coarse-grained location permission can only access location with both low granularity and frequency.

Fig. 9 compares the profiling threats (PoI_{total} and $Prof_{cont}$) posed by apps with fine location permissions and those with coarse location permissions. It also plots the distribution of the privacy metrics for apps without location permissions assuming that they accessed location when running. While this might seems oblivious at a first glance, we aim to compare the location-based usage of apps with different location permissions. This allows us to study if the location permissions are effective as a notification mechanism so that users use apps from different places depending on the location permissions.

The apps with fine-grained location permissions exhibited very similar usage pattern to those apps without location access. The users ran the app from the same places regardless of whether they have location permissions or not. We conclude that this notification mechanism does little to alert users on potential privacy threats and has no effect on the app-usage behavior. Similar observations have also been made by others [17].

Almost a half of the apps (Table 1) that request fine-grained location permissions are found to be able to achieve the location-based functionality with coarser-granularity location. This suggests that apps abuse location permissions. If used appropriately, permissions can be effective in thwarting the threats resulting from apps' abuse of location access (∼40% of the apps — Group B — according to Fig. 8).

Background Location Access: Background location access is critical when it comes to tracking individuals. It enables comprehensive access to the user's mobility information including PoIs and frequent routes. Recently, iOS 8 introduced a new location permission that allows users to authorize location access in the background for apps on their devices.

This permission strikes a balance between privacy and QoS. We showed in Section 4 that apps rarely access location in the background. Thus, this option affects a very low portion of the user's apps, but is effective in terms of privacy protection, especially in thwarting tracking threats. We evaluated the tracking threat in terms of tracking time per day [12, 22] for the three datasets for foreground location access.

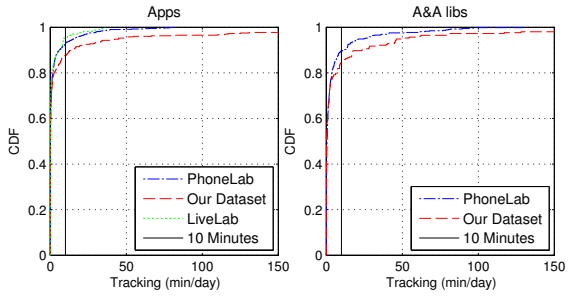

Figure 10: The distribution of the tracking threat posed by the foreground apps (left) and A&A libraries (right).

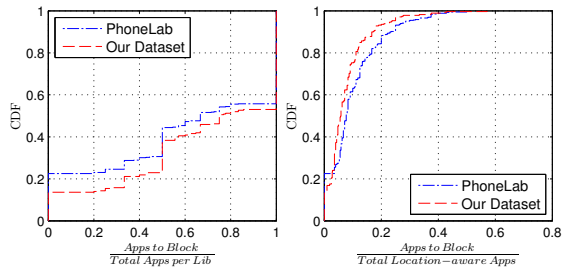

Figure 11: The fraction of the user's apps that must be blocked from accessing location to protect against privacy threats posed by A&A libraries.

Fig. 10 (left) shows that in 90% of the app–user combinations, blocking background location access will limit the location exposure to less than 10 minutes a day (from foreground location access). The third-party libraries tend to pose slightly higher tracking threats than apps (Fig. 10 – right).

Per-app Location Permissions: To improve over static permissions, iOS enables the user to allow/disallow location access on a per-app basis. The users gain two advantages from this model: (i) location access can be blocked for a subset, but not all, of the apps, and (ii) the apps retain some functionality even when the location access is blocked.

Even if the user trusts an app with location access, the app can still profile him through the places he visited (Groups D and E in Fig. 8). To combat these threats, the user has to either allow location access to fully exploit the app and lose privacy, or gain his privacy while losing the location-based app functionality. Currently, mobile platforms offer no middle ground to balance privacy and QoS requirements.

In Section 7, we showed that A&A libraries pose significant threats that users are completely unaware of as they access location from more than one app. The user can't identify which apps he must disallow to access location in order to mitigate threats from third-party libraries. Fig. 11 shows the portion of the user's apps that must be disbarred from accessing location to thwart threats from packed A&A libraries. It turns out (left plot of Fig. 11) that in order to protect the user from privacy threats posed by a single library, at least 50–70% of the apps carrying the library must be disbarred from accessing location. This amounts to blocking location for more than 10% of the apps installed on the device.

In conclusion, a static permission model suffers serious limitations, blocking location access in the background is effective in mitigating the tracking threat but not the profiling one, and per-app controls exhibit an unbalanced tradeoff between privacy and QoS. Also, they are ineffective against the threats caused by A&A libraries. Thus, a finer-grained location access control is required, allowing control for each app session depending on the context. Per-session location access control allows users to leverage better and more space in the privacy–QoS spectrum.

9 LP-Doctor

Users can't utilize the existing controls to achieve per-session location-access controls for two reasons. First, these controls are coarse-grained (providing only per-app controls at best). For finer-level controls, the user has to manually modify the location settings before launching each app, which is quite cumbersome and annoying. Second, even if the user can easily change these settings, making an informed decision is a different story. Therefore, we propose `LP-Doctor` that helps users utilize the existing OS controls to provide location-access control on a per-session basis.

9.1 Design

`LP-Doctor` trusts the underlying OS and its associated apps; it targets user-level apps accessing location while running in the foreground, as we found that most apps don't access location in the background. `LP-Doctor` focuses on the apps with fine location permissions as they could pose higher threats. `LP-Doctor` automatically coarsens location for apps requesting coarse location permissions to ensure a commensurate privacy-protection level.

The main operation of `LP-Doctor` consists of two parts. The first involves the user-transparent operations described below, while the second includes the interactions with the user described in Section 9.2.

We bundled `LP-Doctor` with CyanogenMod's app launcher.[2] It runs as a background service, intercepts app-launch events, decides on the appropriate actions, performs these actions, and then instructs the app to launch. Fig. 12 shows the high-level execution flow of

[2]Source code: `https://github.com/kmfawaz/LP-Doctor`.

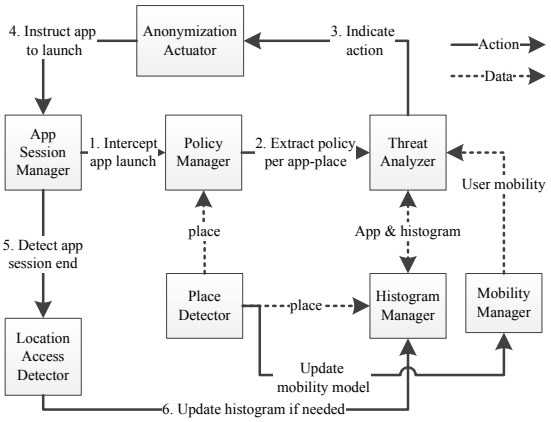

Figure 12: The execution flow of LP-Doctor when a location-aware app launches.

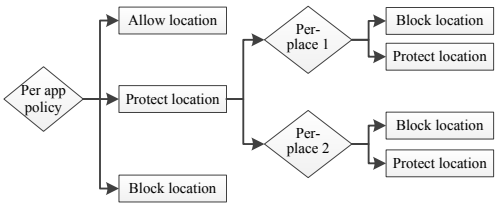

Figure 13: The policy hierarchy of LP-Doctor.

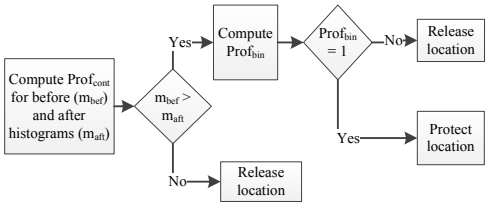

Figure 14: The threat analyzer's decision diagram.

LP-Doctor. Next, we elaborate on LP-Doctor's components and their interactions.

App Session Manager: is responsible for monitoring app launch and exit events. LP-Doctor needs to intercept app-launch events to anonymize location.

Fortunately, Android (recently iOS as well) allow for developing custom app launchers. Users can download and install these launchers from the app store which will, in turn, be responsible for listening to the user's events and executing the apps. We instrumented CyanogenMod's app launcher (available as open source and under Apache 2 license) to intercept app launch events.

Particularly, before the app launcher instructs the app to execute, we stop the execution, save the state, and send an intent to LP-Doctor's background service (step 1 in Fig. 12). LP-Doctor takes a set of actions and sends an intent to the app launcher, signaling the app can launch (steps 2 and 3 in Fig. 12). The app launcher then restores the saved state and proceeds with execution of the app (step 4 in Fig. 12). In Section 9.4, we will report the additional delay incurred by this operation.

In the background, LP-Doctor frequently polls (once every 10s) the current foreground app to detect if the app is still running. For this purpose, it uses `getRecentTasks` on older versions of Android and `AppUsageStats` class for Android L. When an app is no longer running in the foreground, LP-Doctor executes a set of maintenance operations to be described later (steps 5 and 6 in Fig. 12).

Policy Manager: fetches the privacy policy for the currently visited place and the launched app as shown in Fig. 13.

At installation time, the user specifies a privacy policy to be applied for the app. We call this the *per-app policy* which specifies three possible actions: *block*, *allow*, and *protect*. If the per-app policy indicates privacy protection, LP-Doctor asks the user to specify a per-place policy for the app. The per-place policy indicates the policy that LP-Doctor must follow when the app launches from a particular place. The policy manager passes the app's policy and the current place to the threat analyzer.

Place Detector & Mobility Manager: The place detector monitors the user's actual location, and applies online clustering to extract the spatio-temporal clusters which represent places that the users visit. Whenever the user changes the place he is visiting, the place detector module instructs the mobility manager to update the mobility profile of the user as defined in Section 6.

Histogram Manager: maintains the histogram of the places visited as observed by each app. It stores the histograms in an SQLite table that contains the mapping of each app–place combination to a number of observations. The threat analyzer module consults the histogram manager to obtain two histograms whenever an app is about to launch. The first is the current histogram of the app (based on previous app events) which we refer to as the "before" histogram. While the second one is the potential histogram if the app were to access location from the currently visiting place; we call this histogram as the "after" one.

Threat Analyzer: decides on the course of action regarding apps associated with a *protect* policy. It basically performs the decision diagram depicted in Fig. 14 to decide whether to release the location or add noise.

The threat analyzer determines whether the "after" histogram leaks more information than the old one through computing $Prof_{cont}$ for each histogram. If $Prof_{cont}$ increases LP-Doctor decides to release the lo-

cation to the app. On the other hand, if $Prof_{cont}$ decreases, LP-Doctor uses $Prof_{bin}$ to decide if the "after" histogram fits the user's mobility pattern and whether to release or anonymize location.

$Prof_{bin}$ depends on the significance level, α, as we specified in Section 6. In LP-Doctor, α is a function of the privacy level chosen by the user. LP-Doctor recognizes three privacy levels: low, medium, and high. Low privacy corresponds to $\alpha = 0.1$; medium privacy corresponds to $\alpha = 0.05$; and high privacy protection corresponds to the most conservative $\alpha = 0.01$.

The procedure depicted in Fig. 14 won't hide places that the user seldom visits but are sensitive to him. The per-place policies allow the user to set a privacy policy for each visited place, effectively allowing him to control the places he wants revealed to the service providers. Also, LP-Doctor can be extended to support other privacy criteria that try to achieve optimal privacy by perturbing location data [9, 37].

Anonymization Actuator: receives an action to perform from the threat analyzer. If the action is to protect the current location, the actuator computes a fake location by adding Laplacian noise [3] to ensure location indistinguishability. The privacy level determines the amount of noise to be added on top of the current location. One the other hand, if the action is to block, the actuator computes the fake location of $< 0, 0 >$.

As specified by Andrés *et al.* [3], repetitive engagement of Laplacian noise mechanism at the same location leaks information about the location. To counter this threat, LP-Doctor computes the anonymized location once per location and protection-level combination, and saves it. When the user visits the same location again, LP-Doctor employs the same anonymized location that was previously computed to prevent LP-Doctor from recomputing a fake location for the same place.

After computing/fetching the fake location, the actuator module will engage the mock location provider. The mock location provider is an Android developer feature to modify the location provided to the app from Android. It requires no change in the OS or the app. The actuator then displays a non-intrusive notification to the user, and signals the session manager to start the app.

End-of-Session Maintenance: When the app finishes execution, the actuator disengages the mock location provider, if engaged. The location-access detector will then detect if the app accessed location to update the app's histogram accordingly. The location access detector performs a "dumpsys location" to exactly detect if the app accessed location or not while running. If it did access location, the location-access detector module updates the app's histogram (increment the number of visits from the current location). It is worth noting that LP-Doctor treats sessions of the same app within 1 min as the same app session.

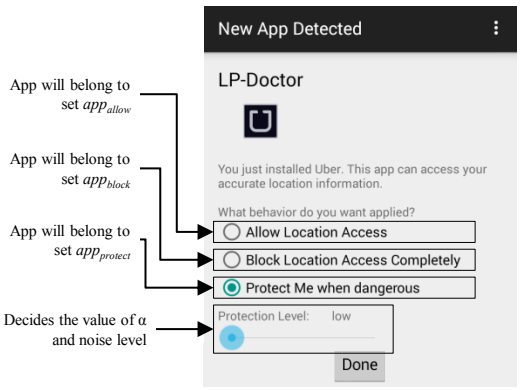

Figure 15: The installation menu.

9.2 User Interactions

LP-Doctor interacts with the user to communicate privacy-protection status. It also enables him to populate the privacy profiles for different apps and places. As will be evident below, the main philosophy guiding LP-Doctor's design is to minimize the user interactions, especially intrusive ones. We satisfy two design principles proposed by Felt *et al.* [13] that should guide the design of a permission granting UI. The first principle is to conserve user attention by not issuing excessively repetitive prompts. The second is to avoid interrupting the user's primary tasks.

Bootstrapping Menu: The first communication instance with LP-Doctor takes place upon its installation. LP-Doctor will ask the user to set general configuration options. These options include (1) alerting the user when visiting a new location to set the per-place policies and (2) invoking protection for A&A libraries. The menu will also instruct the user to enable the mock location provider and grant the app "DUMP" permissions through ADB. This interaction takes place only once per LP-Doctor's lifetime.

Installation Menu: LP-Doctor prompts the user when a new (non-system and location-aware) app is installed. The menu enables the user to set the per-app profiles. Fig. 15 shows the displayed menu when an app ("uber" in this case) has finished installation. The user can choose one of three options which populates three app sets: app_{allow}, app_{block}, and $app_{protect}$.

Logically, this menu resembles the per-app location settings for iOS, except that it provides users with an additional option of privacy protection. The protection option acts as a middle-ground between completely allowing and blocking location access to the app. The user will interact with this menu; only once per app, and only for non-system apps that requests the fine location permission. Based on our PhoneLab dataset, we estimate

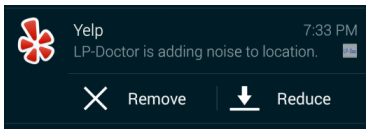

Figure 16: LP-Doctor's notification when adding noise.

that the user will be issued this menu on average for one app he installs per five installed apps on the device.

Per-Place Prompts: LP-Doctor relies on the user to decide its actions in different visited places, if he agrees to get prompted when visiting new places. Specifically, whenever the user visits a new place, LP-Doctor prompts him to decide on actions to perform when running apps that the user chose to protect. We call these *per-place policies* (Fig. 13).

The per-place policies apply for apps belonging to the set $app_{protect}$. The user has the option to specify whether to block location access completely, or apply protection. Applying protection will proceed to execute the operations of the threat analyzer as defined in Fig. 14. LP-Doctor allows the user to modify the policies for each app–place combination.

LP-Doctor issues this prompt only when the user launched an app of the set $app_{protect}$ from a new location. From our PhoneLab dataset, we estimate that such a prompt will be issued to the user at most once a week.

Notifications: As specified earlier, the threat actuator displays a non-intrusive notification (Fig. 16) to the user to inform him about the action being taken.

If the action is to allow location access (because the policy dictates so or there is no threat), LP-Doctor notifies the user that there is no action being taken. The user has the option to invoke privacy protection for the current app session. If the user instructs LP-Doctor to add noise for a single app over two consecutive sessions from the same place, LP-Doctor will create a per-place policy for the app and move it to the $app_{protect}$ set if it were part of app_{allow}.

On the other hand, if LP-Doctor decides to add noise to location or block it, it will notify the user of it (Fig. 16). The notification includes two actions that the user can make: remove or reduce noise. If the user overrides LP-Doctor's actions for two consecutive sessions of an app from the same place, LP-Doctor remembers the decision for future reuse.

LP-Doctor leverages the user's behavior to learn the protection level that achieves a favorable privacy–utility tradeoff. Since the mapping between the chosen privacy and noise levels is independent of the running app, the functionality of certain apps might be affected. LP-Doctor allows the user to fine-tune this noise level and then remembers his preference for future reuse.

Reducing the noise level will involve recomputing the fake location with a lower noise value (if no such location has been computed before). One could show that leak of information (from lowering noise level successively) will be capped by that corresponding to the fake location with the lowest noise level released to the service provider.

Using our own and PhoneLab's datasets, we estimate LP-Doctor's need to issue such non-intrusive notification (indicating protection taking place) for only 12% of the sessions on average for each app.

9.3 Limitations

The user-level nature of LP-Doctor introduces some limitations related to certain classes of apps. First, LP-Doctor, like other mechanisms, is inapplicable to apps that require accurate location access such as navigation apps for elongated period of times.

Second, LP-Doctor can't protect the user against apps utilizing unofficial location sources such as "WeChat." Such apps might scan for nearby WiFi access points and then use scan results to compute location. LP-Doctor can't anonymize location fed to such apps, though it can warn the user of the privacy threat incurred if the user is to invoke the location-based functionality. Also, it can offer the user the option to turn off the WiFi on the device to prevent accurate localization by the app when running.

Finally, LP-Doctor doesn't apply privacy protection to the apps continuously accessing location while running in the background. Constantly invoking the mock location provider affects the usability of apps that require fresh and accurate location when running. Fortunately, we found that the majority of the apps don't access location in the background (Section 4). Nevertheless, this still highlights the need for OS support to control apps' location access in the background (like the one that iOS currently provides).

9.4 Evaluation

We now evaluate and report LP-Doctor's overhead on performance, Quality of Service (QoS), and usability.

9.4.1 Performance

LP-Doctor performs a set of operations which delay the app launching. We evaluate this delay on two devices: Samsung Galaxy S4 running Android 4.2.2, and Samsung Galaxy S5 running Android 4.4.4. We recorded the delay in launching a set of apps while running LP-Doctor. We partitioned those apps into two sets. The first (set 1) includes the apps which LP-Doctor doesn't target, while the second (set 2) includes non-system apps that request fine location permissions.

Fig. 17 plots the delay distribution for both devices and for the two app sets. Clearly, apps that belong to the

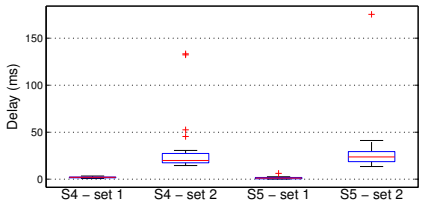

Figure 17: The app launch delay caused by LP-Doctor.

first set experience very minimal delay, varying between 1 and 3ms. The second set of apps experience longer delays without exceeding 50ms for both devices. We also tested LP-Doctor's impact on the battery by recording the battery depletion time when LP-Doctor was running in the background and when it was not. We found that LP-Doctor has less than 10% energy overhead (measured as the difference in battery depletion time). Besides, LP-Doctor runs the same logic as our PhoneLab survey app in the background which 95 users ran over 4 months and reported no performance or battery issues.

9.4.2 User Study

To evaluate the usability of LP-Doctor and its effect on QoS, we conducted a user study over Amazon Mechanical Turk. We designed two Human Intelligence Tasks (HITs), each evaluating a different representative testing scenario of LP-Doctor.

Apps that provide location-based services (LBSes) fall into several categories. On one dimension, an app can *pull* information to the user based on the current location, or it can *push* the user's current location to other users. On another dimension, the app can access the user's location *continuously* or *sporadically* to provide the LBS. One can then categorize apps as: *pull-sporadic* (e.g., weather, Yelp, etc.), *pull-continuous* (e.g., Google Now), *push-sporadic* (e.g., geo-tagging, Facebook check-in, etc.), or *push-continuous* (e.g., Google Latitude). As LP-Doctor isn't effective against apps continuously accessing the user's location (which are a minority to start with), we focus on studying the user's experience of LP-Doctor while using **Yelp**, as a representative example of *pull-sporadic* apps, and **Facebook**, as representative example of *push-sporadic* apps.

We recruited 120 participants for the Yelp HIT and another 122 for the Facebook HIT[3]; we had 227 unique participants in total. On average, each participant completed the HIT in 20min and was compensated $3 for his response. We didn't ask the users for any personal information and nor did LP-Doctor . We limited the study to Android users.

Of the participants: 28% were females vs. 72% males; 32% had high school education, 47% with BS degree or equivalent; and 37% are older than 30 years. Also, 52% of the participants reported that they have taken steps to mitigate privacy threats. Interestingly, 93% of the participants didn't have mock locations enabled on their devices indicating the participants are not tech-savvy.

We constructed the study with a set of connected tasks. In every task, the online form displays a set of instructions/questions that the participant user must follow/answer. After successfully completing the task, LP-Doctor displays a special code that the participant must input to proceed to the next task. In what follows, we describe the various tasks that we asked users to perform and how they responded.

Installing and configuring LP-Doctor: The participants' first task was to download LP-Doctor from Google Play and enable mock locations. We asked the users to rate how difficult it was to enable mock locations on the scale of 1 (easy) to 5 (difficult). 83% of the participants answered with a value of 1 or 2 implying that LP-Doctor is easy to install.

Installation menu: In their second task, the participants interacted with the installation menu (Fig. 15). The users had to install (re-install if already installed) either Yelp or Facebook. Just when either app completes installation, LP-Doctor presents the user with the menu to input the privacy options. The participants reported a positive experience with this menu; 83% reported it was easy to use (rated 1 or 2 on a scale of 1 (easy) to 5 (hard)); 86% said it was informative; 83% thought it provides them with more control than Android's permission; 79% answered it is useful (rated 1 or 2 on a scale of 1 (useful) to 5 (useless)); and 74% would like to have such menu appearing whenever they install a location-aware app (12% answered with not sure).

Impact on QoS: The survey version of LP-Doctor adds noise on top of the user's location regardless of his previous choice. This allowed us to test the impact of adding noise (Laplacian with 1000m radius) to the location accessed by either Yelp or Facebook. We didn't ask the participants to assess the effect of location anonymization on the QoS directly. Rather, we asked the Yelp respondents to report their satisfaction with the list of restaurants returned by the app. While we asked the Facebook respondents to indicate whether the list of places to check-in from is relevant to them. The participants in the first HIT indicated that Yelp ran normally (82%), the restaurant search results were relevant (73%), the user experience didn't change (76%), and Yelp need not access the user's accurate location (67%).

The Facebook HIT participants exhibited similar results: Facebook ran normally (80%), the list of places to check-in was relevant (60%), user experience didn't change (80%), and Facebook need not access the user's

[3] https://kabru.eecs.umich.edu/wordpress/wp-content/uploads/lp-doctor-survey-fb.pdf

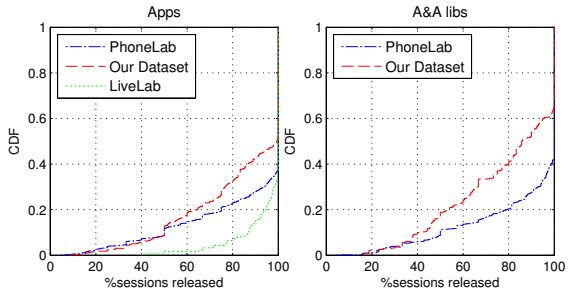

Figure 18: The distribution of percentage of sessions where apps maintain QoS for apps (left) and A&A libraries (right).

accurate location (80%).

Fig. 18 shows the percentage of sessions (for all app–user combinations) that won't experience any noise addition according to our datasets. It is obvious that the percentage of sessions with potential loss in QoS (when `LP-Doctor` adds noise) is minimal (less than 20%, a bit higher if the user opts for A&A libraries protection). Our user study shows that more than 70% of the users won't experience loss in QoS in these sessions. For those users who do face loss in QoS, `LP-Doctor` provides them with the option of adjusting the noise level at runtime through the notifications.

Notifications: In the final task, we asked the participants to test the noise reduction feature that allows for a personalized privacy–utility trade-off. After they reduced the noise level, they would invoke the location-based feature in both Yelp and Facebook and check if the results were improved. Indeed, most of the participants who reported loss in QoS reported the Yelp's search results (64%) and Facebook's check-in places (70%) improved after reducing the noise.

The participants also indicated the the noise reduction feature is easy to use (75%). 86% of the participants won't mind having this feature whenever they launch a location-aware app.

Post-study questions: As we couldn't control the per-place prompts given our study design, we asked the participants for their opinion about being prompted when visiting new places (per-place prompts). Only 54% answered they would prefer prompted, 37% answered negatively, and the rest answered "I am not sure." These responses are consistent with our design decision; the user has to approve per-place prompts when initially configuring `LP-Doctor` as they are not automatically enabled.

Also, 82% of the participants felt comfortable that Facebook (80%) and Yelp (85%) didn't access their accurate location. Finally, 77% of the participants answered "Yes" when asked about installing `LP-Doctor` or other tool to protect their location privacy. Only 11% answered "No" and the rest answered with "I am not sure." This result comes at an improvement over the 52% who initially said they took steps in the past to address location-privacy threat.

In summary, we conducted one of the few studies (e.g., [8]) that evaluate a location-privacy protection mechanism in the wild. We showed that location-privacy protection is feasible in practice where a balance between QoS, usability, and privacy could be achieved.

10 Conclusion

In this paper, we posed a question about the effectiveness of OS-based location-access controls and whether they can be improved. To answer this question, we conducted a location-collection campaign that considers location-privacy threats from the perspective of mobile apps. From this campaign, we observed, modeled, and categorized profiling as being the prominent privacy threat from location access for both apps and A&A libraries. We concluded that controlling location access per session is needed to balance between loss in QoS and privacy protection. As existing OS controls don't readily provide such functionality, we proposed `LP-Doctor`, a user-level tool that helps the user better utilize existing OS-based location-access controls. `LP-Doctor` is shown to be able to mitigate privacy threats from both apps and A&A libraries with little effect on usability and QoS. In future, we would like to test `LP-Doctor` in the wild and use it to explore the dynamics that affect users' decisions to install a location-privacy protection mechanism.

11 Acknowledgments

We would like to thank the anonymous reviewers and the shepherd, Reza Shokri, for constructive suggestions. The work reported in this paper was supported in part by the NSF under Grants 0905143 and 1114837, and the ARO under W811NF-12-1-0530.

References

[1] Almuhimedi, H., Schaub, F., Sadeh, N., Adjerid, I., Acquisti, A., Gluck, J., Cranor, L. F., and Agarwal, Y. Your location has been shared 5,398 times!: A field study on mobile app privacy nudging. In *Proceedings of CHI '15* (2015), pp. 787–796.

[2] Amini, S., Lindqvist, J., Hong, J., Lin, J., Toch, E., and Sadeh, N. Caché: Caching location-enhanced content to improve user privacy. In *Proceedings of MobiSys '11* (New York, NY, USA, 2011), ACM, pp. 197–210.

[3] Andrés, M. E., Bordenabe, N. E., Chatzikokolakis, K., and Palamidessi, C. Geo-indistinguishability: Differential privacy for location-based systems. In *Proceedings of CCS '13*.

[4] ANDRIENKO, G., GKOULALAS-DIVANIS, A., GRUTESER, M., KOPP, C., LIEBIG, T., AND RECHERT, K. Report from dagstuhl: the liberation of mobile location data and its implications for privacy research. *SIGMOBILE Mob. Comput. Commun. Rev. 17*, 2 (July 2013), 7–18.

[5] ASHFORD, W. Free mobile apps a threat to privacy, study finds. http://www.computerweekly.com/news/2240169770/Free-mobile-apps-a-threat-to-privacy-study-finds, October 2012.

[6] BERESFORD, A. R., RICE, A., SKEHIN, N., AND SOHAN, R. Mockdroid: Trading privacy for application functionality on smartphones. In *Proceedings of HotMobile '11* (New York, NY, USA, 2011), ACM, pp. 49–54.

[7] BETTINI, C., WANG, X., AND JAJODIA, S. Protecting privacy against location-based personal identification. *Secure Data Management* (2005), 185–199.

[8] BILOGREVIC, I., HUGUENIN, K., MIHAILA, S., SHOKRI, R., AND HUBAUX, J.-P. Predicting Users' Motivations behind Location Check-Ins and Utility Implications of Privacy Protection Mechanisms. In *NDSS'15* (2015).

[9] BORDENABE, N. E., CHATZIKOKOLAKIS, K., AND PALAMIDESSI, C. Optimal geo-indistinguishable mechanisms for location privacy. In *Proceedings of CCS '14* (2014), pp. 251–262.

[10] CLAUSET, A., SHALIZI, C. R., AND NEWMAN, M. E. J. Power-law distributions in empirical data. *SIAM Rev. 51*, 4 (Nov. 2009), 661–703.

[11] DE MONTJOYE, Y.-A., HIDALGO, C. A., VERLEYSEN, M., AND BLONDEL, V. D. Unique in the crowd: The privacy bounds of human mobility. *Sci. Rep. 3* (Mar 2013).

[12] FAWAZ, K., AND SHIN, K. G. Location privacy protection for smartphone users. In *Proceedings of CCS '14* (New York, NY, USA, 2014), ACM, pp. 239–250.

[13] FELT, A. P., EGELMAN, S., FINIFTER, M., AKHAWE, D., AND WAGNER, D. How to ask for permission. In *Proceedings of HotSec'12* (2012).

[14] FISHER, D., DORNER, L., AND WAGNER, D. Short paper: Location privacy: User behavior in the field. In *Proceedings of SPSM '12* (2012), pp. 51–56.

[15] FREUDIGER, J., SHOKRI, R., AND HUBAUX, J.-P. Evaluating the Privacy Risk of Location-Based Services. In *Financial Cryptography and Data Security (FC)* (2011).

[16] FRITSCH, L. Profiling and location-based services (lbs). In *Profiling the European Citizen*, M. Hildebrandt and S. Gutwirth, Eds. Springer Netherlands, 2008, pp. 147–168.

[17] FU, H., YANG, Y., SHINGTE, N., LINDQVIST, J., AND GRUTESER, M. A field study of run-time location access disclosures on android smartphones. In *Proceedings of USEC 2014*.

[18] GOLLE, P., AND PARTRIDGE, K. On the anonymity of home/work location pairs. In *Proceedings of PERVASIVE '09* (Berlin, Heidelberg, 2009), Springer-Verlag, pp. 390–397.

[19] GOODWIN, C. A conceptualization of motives to seek privacy for nondeviant consumption. *Journal of Consumer Psychology 1*, 3 (1992), 261 – 284.

[20] GUHA, S., JAIN, M., AND PADMANABHAN, V. N. Koi: A location-privacy platform for smartphone apps. In *Proceedings of NSDI'12* (2012), USENIX Association, pp. 14–14.

[21] HIGGINS, E. T. Self-discrepancy: a theory relating self and affect. *Psychological Review 94*, 3 (Jul 1987), 319–340.

[22] HOH, B., GRUTESER, M., XIONG, H., AND ALRABADY, A. Achieving guaranteed anonymity in gps traces via uncertainty-aware path cloaking. *IEEE TMC 9*, 8 (August 2010), 1089–1107.

[23] HORNYACK, P., HAN, S., JUNG, J., SCHECHTER, S., AND WETHERALL, D. These aren't the droids you're looking for: retrofitting android to protect data from imperious applications. In *Proceedings of CCS '11* (2011), pp. 639–652.

[24] JUNG, J., HAN, S., AND WETHERALL, D. Short paper: Enhancing mobile application permissions with runtime feedback and constraints. In *Proceedings of SPSM '12* (2012), pp. 45–50.

[25] KRUMM, J. Inference attacks on location tracks. In *Proceedings of PERVASIVE '07* (2007), Springer-Verlag, pp. 127–143.

[26] KRUMM, J. Realistic driving trips for location privacy. In *Proceedings of PERVASIVE '09* (2009), Springer-Verlag, pp. 25–41.

[27] KULLBACK, S., AND LEIBLER, R. A. On information and sufficiency. *Ann. Math. Statist. 22*, 1 (03 1951), 79–86.

[28] LU, H., JENSEN, C. S., AND YIU, M. L. Pad: privacy-area aware, dummy-based location privacy in mobile services. In *Proceedings of MobiDE '08* (2008), pp. 16–23.

[29] MEYEROWITZ, J., AND ROY CHOUDHURY, R. Hiding stars with fireworks: location privacy through camouflage. In *Proceedings of MobiCom '09* (2009), pp. 345–356.

[30] MICINSKI, K., PHELPS, P., AND FOSTER, J. S. An Empirical Study of Location Truncation on Android. In *Mobile Security Technologies (MoST '13)* (San Francisco, CA, May 2013).

[31] NANDUGUDI, A., MAITI, A., KI, T., BULUT, F., DEMIRBAS, M., KOSAR, T., QIAO, C., KO, S. Y., AND CHALLEN, G. Phonelab: A large programmable smartphone testbed. In *Proceedings of SENSEMINE'13* (2013), pp. 4:1–4:6.

[32] PALANISAMY, B., AND LIU, L. Mobimix: Protecting location privacy with mix-zones over road networks. In *ICDE 2011* (april 2011), pp. 494 –505.

[33] PINGLEY, A., ZHANG, N., FU, X., CHOI, H.-A., SUBRAMANIAM, S., AND ZHAO, W. Protection of query privacy for continuous location based services. In *INFOCOM'11* (April 2011), IEEE.

[34] SHEPARD, C., RAHMATI, A., TOSSELL, C., ZHONG, L., AND KORTUM, P. Livelab: measuring wireless networks and smartphone users in the field. *SIGMETRICS Perform. Eval. Rev. 38*, 3 (Jan. 2011), 15–20.

[35] SHOKRI, R., THEODORAKOPOULOS, G., DANEZIS, G., HUBAUX, J.-P., AND LE BOUDEC, J.-Y. Quantifying location privacy: the case of sporadic location exposure. In *Proceedings of PETS'11* (2011), pp. 57–76.

[36] SHOKRI, R., THEODORAKOPOULOS, G., LE BOUDEC, J., AND HUBAUX, J. Quantifying location privacy. In *Security and Privacy (SP), 2011 IEEE Symposium on* (may 2011), pp. 247 –262.

[37] SHOKRI, R., THEODORAKOPOULOS, G., TRONCOSO, C., HUBAUX, J.-P., AND LE BOUDEC, J.-Y. Protecting location privacy: Optimal strategy against localization attacks. In *Proceedings of CCS '12* (2012), pp. 617–627.

[38] THURM, S., AND KANE, Y. I. Your apps are watching you. http://online.wsj.com/article/SB10001424052748704694004576020083703574602.html, December 2010.

[39] TRIPP, O., AND RUBIN, J. A bayesian approach to privacy enforcement in smartphones. In *USENIX Security 14* (San Diego, CA, 2014), USENIX Association, pp. 175–190.

[40] VAJNA, S., TTH, B., AND KERTSZ, J. Modelling bursty time series. *New Journal of Physics 15*, 10 (2013), 103023.

[41] ZANG, H., AND BOLOT, J. Anonymization of location data does not work: a large-scale measurement study. In *Proceedings of MobiCom '11* (New York, NY, USA, 2011), ACM, pp. 145–156.

[42] ZICKUHR, K. Location-based services. http://pewinternet.org/Reports/2013/Location.aspx, September 2013.