Location Privacy Breach: Apps Are Watching You in Background

Dachuan Liu
College of William and Mary
Email: dliu@cs.wm.edu

Xing Gao
College of William and Mary
Email: xinggao@cs.wm.edu

Haining Wang
University of Delaware
Email: hnw@udel.edu

***Abstract*—Smartphone users can conveniently install a set of apps that provide Location Based Service (LBS) from markets. These LBS-based apps facilitate users in many application scenarios, but they raise concerns on the breach of privacy related to location access. Smartphone users can hardly perceive location access, especially when it happens in background. In comparison to location access in foreground, location access in background could result in more serious privacy breach because it can continuously know a user's locations. In this paper, we study the problem of location access in background, and especially perform the first measurement of this background action on the Google app market. Our investigation demonstrates that many popular apps conduct location access in background within short intervals. This enables these apps to collect a user's location trace, from which the important personal information, Points of Interest (PoIs), can be recognized. We further extract a user's movement pattern from the PoIs, and utilize it to measure the extent of privacy breach. The measurement results also show that using the combination of movement pattern related metrics and the other PoI related metrics can help detect the privacy breach in an earlier manner than using either one of them alone.**

I. Introduction

With the prosperity of smartphone markets [29], Location Based Service (LBS) has been widely applied to facilitate smartphone users. LBS fits the mobile platform very well since real-time location data can be easily acquired via the GPS or networking components on a mobile device. Usually, the location of a smartphone indicates the place where its owner stays. With the help of LBS, smartphone users can benefit in many application scenarios, e.g., finding a restaurant or getting promotions of merchandises nearby. However, the location access also raise concerns on privacy breach, which has been revealed in many previous studies [18], [23], [24], [34], [35], [36].

On a mobile platform, the system resources including user location are managed and protected by OS. As one of the most popular mobile OSes, Android protects user location via its permission mechanism, which requires apps to declare a permission before they are able to access the corresponding resource. Users can grant or revoke the permission based on specific requirements. Although such a mechanism allows users to make the decision on location access, it is quite difficult for users to determine whether a location access should be granted or not. Users rarely notice the action of location access even though Android displays a notification on its system bar [1]. There are some solutions, such as [11], [12], proposed to help a user manage location access. However, these works focus on handling location access in foreground. Apps collecting location in foreground get the data when users interact with their UIs. These locations are usually discrete and may not be the places that users are really interested in [11]. Differently, location access in background allows apps to continuously access locations. The location trace can be used to infer the Points of Interest (PoIs) where users have stayed for some time. The PoIs reveal more private personal information about users, and the patterns of how users visit these places could further reflect the habituation of users.

In this paper, we study the privacy breach caused by location access in background in the popular app market. First, we examine the top 100 popular apps of 28 categories on the Google app market. Our measurements show that 1,140 out of 2,800 apps declare the permission for location access. To our surprise, about 9% of the 1,140 apps access location in background. These apps demonstrate different characteristics in the access process, i.e., they request locations at different granularities and frequencies, which could lead to the privacy leakage to different extents. Then, we measure the potential privacy risks based on several widely used privacy metrics. We leverage the Spatio-Temporal algorithm [3] to identify PoIs from the location traces of users. We further extract the movement pattern ($PoI_i \rightarrow PoI_j$) which implies the user's habituation in daily activities, and use it to measure the privacy risk incurred by location access. In comparison to the existing method of using the ⟨region, visited times⟩, using ⟨movement pattern, happen times⟩ can bring earlier detection of privacy breach in many cases.

The remainder of the paper is organized as follows. In Section II, we survey the related work. The statistical analysis of location access in background in the current app market is described in Section III. Then, we present the threat model, the privacy metrics, and the measurements based on the characteristics of apps and real location data traces in Section IV. Finally, we conclude this work in Section V.

II. Related work

Many research works have revealed that collected locations can be exploited to infer a user's private information. Bettini et al. [6] first pointed out that the spatial and temporal correlation could be leveraged as the quasi-identifier, with which an adversary can infer a user's identity from the anonymized location

data. By analyzing a large amount of cellular records, Zang et al. [35] showed that anonymized location data still contains a user's private information, and the top 2 and 3 locations could yield a very small anonymous user set, leading to an easy guess. In the work [14], the authors defined three types of locations to infer a user's identity. Montjoye et al. [7] demonstrated that a few number of Spatio-Temporal points can be used to uniquely identify most individuals. Coarsening the locations cannot guarantee a user to be anonymized. The authors in [33] indicated that users are not quite clear about resource accesses from apps on Android. More than that, users rarely recognize location access when it happens, although Android prompts users by displaying a mark on the system bar [1].

We may not expect that the protection mechanisms of Android can handle the risks in various emerging applications. So, researchers have proposed many complementary solutions to address the specific problems. TaintDroid, AppFence, and PmDroid [10], [25], [16] leverage the dynamic analysis technique to detect information leakage in Android systems. Although they can prevent the location from being sent out of the device, they cannot determine whether releasing a location to apps will cause the privacy breach or not. MockDroid [5] first extends Android's permission management so that users can make different choices (releasing real or fake data to an app or blocking the access from an app). As a more advanced solution, TISSA [38] can send shadow data of several kinds of private information. Nauman et al. [28] proposed Apex, a policy enforcement framework, to execute customized policies of using system resources. These solutions depend on users' decisions to work. However, making a proper decision is not easy for users.

Previous research works present some privacy metrics and algorithms for measuring the privacy risk of releasing locations. The spatial cloaking algorithms [17], [20], [31] reduce the accuracy of locations to satisfy the k-anonymity constraint. Some other algorithms [4], [22], [32] protect privacy by selectively hiding locations at an interval or in high density areas. Hoh et al. [23], [24] proposed a different metric called time to confusion that measures the privacy according to how long an adversary needs to track a user before identifying the user. Shokri et al. [30] presented the location information disclosure attacks, based on which they proposed a framework to measure the performance of different Location Privacy Protection Mechanisms (LPPMs). Entropy and k-anonymity are used as the metrics in privacy measurement. These solutions are deployed on a trust server that manages the location data of all nearby users. They are not suitable for smartphone users, because the trust server does not know what information the LBS provider has. In this work, we measure the problem based on the data of a single user.

There are a couple of works aiming to tackle the privacy issues of location access on Android. Micinski et al. [27] evaluated the location truncation technique that reduces the risk of leaking a user's privacy while apps can normally work with them in many cases. In the work [15], the authors studied the location access of running apps by checking whether the cached location is changed. Their method does not apply for checking location access in background. LP-Guardian [12] simply releases the coarsened location to requests from background apps. Fawaz et al. [11] conducted the first location measurement campaign by analyzing the popular apps. Their privacy model mainly works for those cases where apps sporadically access location in foreground. Our work is different from the above works because we measure the risk of location access in background based on the metrics integrated with its characteristics.

III. Location Access in Background of Apps

On Android, only one app can run activity on the top of screen. When a user clicks the home button or launches another app, the previous app will be moved to background and cached so that it can be quickly switched to foreground next time. Usually, apps request location per user's request in foreground. However, apps can also receive updates of location in background by registering a location listener. They can even create a persistent service to continuously update locations. Although users could be aware of the action by observing the notification on the system bar when an app accesses location, it is very difficult to recognize the action when it happens in background. Even worse, users may mistake that the location access from a background app is from the foreground app.

In comparison to location access in foreground, location access in background happens in a more concealed manner, and can collect more locations of users. Therefore, it could incur more serious privacy risks. Here we attempt to characterize such an action in the app market.

A. Apps source and study method

As one of the largest app markets, the Google app market provides abundant apps that are classified in 28 categories. We conduct our measurement based on the top 100 popular apps from every category to cover all types of apps. There are 2,800 apk files of the apps downloaded in total. In this work, we detect location access by checking the official methods of requesting location. Note that some apps may not request location via system API. They could parse location by exploiting the cellular or network information. So, location access in background in the app market could be more serious than what we have measured. The problem of detecting location access via unofficial channels is complementary to this work.

First, we separate the apps that do not access location from the downloaded apks. These apps do not declare permissions related to location access. We extract the manifest file from the apk file by using the Apktool [2] for reverse engineering. For those apps that declare at least one permission of location access, we manually install and operate them one by one on a real mobile device (Google Nexus 4 smartphone with Android 4.4 running on it). We launch the app, try to trigger location access, move the app to background, and finally close it. We use a system diagnostic tool "dumpsys" [9] to examine how apps request location. The output of "dumpsys" informs which app

TABLE I: Usage of location provider

Granularity \ Location providers	gps	network	passive	gps network	gps passive	network passive	gps network passive	fused network
Fine	7	3	4	2	0	1	1	0
Coarse	0	0	6	0	0	0	0	0
Fine & Coarse	32	9	7	14	5	4	6	1

is accessing the location, what location provider is registered and how frequently the app requests location.

B. Statistics of the location access

The granularity of location and the frequency of location access directly impact the accuracy and amount of locations that an app can collect. Intuitively, the finer the granularity and the higher the frequency, the more information an app can get. It implies that an adversary has higher possibility to breach a user's privacy. Thus, we mainly observe these two factors of the action (i.e., location access in background) in the measurement.

Overall, there are 1,137 out of 2,800 apps declare at least one permission for location access. Among the 1,137 apps, 17% of them only declare the permission of "ACCESS_FINE_LOCATION" and 16% of them just declare the permission of "ACCESS_COARSE_LOCATION". A majority of the apps (67%) declare both permissions. Those apps, which claim to access fine location, can get locations at either fine or coarse granularity. Android provides four types of location providers: GPS, network, passive, and fused. The first two return fine and coarse locations, respectively. The passive provider allows apps to get the cached location that will be updated if an app requests a new location. Requesting cached location will not induce any extra overhead for location calculation. The fused provider is an interface of Google play service that provides all types of locations to apps. In a real environment, apps may not mean to access location even if they have declared the permission for location access, due to the widely existing problem known as over-privilege [13]. We observe that 528 out of the 1,137 apps function to access location, in which 393 apps automatically start to request location right after they are launched. For the other 135 apps, we trigger the location requests as a normal user.

In many cases, apps access location per user's request. However, there exists location access in background that can hardly be perceived by users. Among the 528 apps that function to access location, 102 out of them (19.3%) will access location in background. Among the 102 apps, 85 apps automatically request location right after they are launched.

Then we investigate the two factors that can lead to the privacy breach to different extents. The first factor depends on the location providers that the apps use. Table I lists the granularity (fine, coarse) of location requested by the apps in manifest files and the location providers (GPS, network, passive, fused) actually used by the apps. There are 96 apps (94.12%) claiming to access fine location, and only 6 apps claiming to access coarse location. However, there exist the cases in which apps do not function as what they claim to do. Our observation shows that 28 (27.5%) apps access coarse location during usage while they claim to access fine location. Meanwhile, we also observe 68 apps (a significant proportion 66.7% of them) access precise location. Considering we may miss some execution paths that can trigger requesting fine location, the above percentage number could be even higher. The coarse locations do not contain much important personal information about the user's privacy. In comparison to releasing coarse location, releasing precise location will incur higher risk of privacy leakage. Some previous solutions have used coarse locations to prevent a user's PoIs from being identified [12], [27].

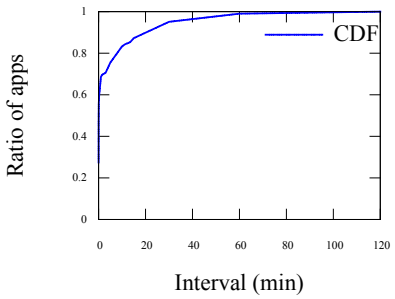

Fig. 1: Cumulative distribution function of location request frequency by apps.

The other factor is the frequency of updating location. For the apps that access location in background, most of them frequently update locations. As shown in Figure 1, 57.8% of the apps update location within every 10 seconds, 68.6% of them do it within every 60 seconds, and 83.8% of the apps keep the location to be updated within every 600 seconds. The largest interval between two location accesses is 7,200 seconds. Only one app requests to update location in that frequency. Releasing location in high frequency enables the app to collect the location trace of a user. The higher the frequency, the more complete the trace is. Usually a complete trace contains much more useful information than an incomplete one. For instance, according to a complete location trace, we can find out the places where a user has stayed for a while. However, using an incomplete location trace, we cannot figure out such places but only know the locations where the user has been. In the next section, we will quantify the potential privacy risk with the change of the frequency.

IV. The Privacy Model and Evaluation

LBS providers can keep the collected location information of users. Even if these information is anonymized by the providers,

TABLE II: Metrics used to measure and quantify the risk.

Metric	Description
PoI_{total}	The fraction of PoIs
$PoI_{sensitive}$	The fraction of user's infrequent visited PoIs
His_{bin}	Whether the histograms of collected data fits the profile
$Deg_{anonymity}$	The degree of anonymity

a third party is still able to restore some private information from the data [35], e.g., inferring a user's identity. In this section, we present our threat model, the privacy metrics that can be used to quantify the privacy risk of a location request, and the risk measurements based on these metrics.

A. Threat model

In our threat model, the adversary is not a malicious entity that aims to exploit the security vulnerabilities of Android systems to steal a user's private information. Instead, it could be any third party, which possesses some of a user's location information, and attempt to infer more private and valuable information so that it can provide a customized service [11], [12]. Either an LBS provider or any third party could be such a entity if it fits the description above. The app deployed on a user's mobile device will collect location traces and send them to LBS providers. Then, the third party could get the anonymized data from LBS providers. The adversary could use various methods to infer the user's private information from the collected location traces. Our assessment of the privacy risk is not based on the specific methods that the adversary uses to infer a user's private information. Instead, we measure the privacy risk in terms of how the released locations demonstrate the PoIs of a user. Intuitively, the more complete location trace the adversary can acquire, the more accurate PoIs of a user can be extracted from the trace.

B. Privacy metrics

Table II lists the four privacy metrics that we use to measure the potential risks. PoIs represent the places where users are interested in and involve some daily activities. They are very important private information since an adversary could infer a user's identity from them. Some PoIs may contain sensitive information of a user, which may bring more serious information leakage. We use two well-known PoI related metrics [11], PoI_{total} and $PoI_{sensitive}$, to measure the risk in this work. The former metric means the total PoIs extracted by the adversary from the collected locations, while the latter metric represents the sensitive PoIs. We count those PoIs where users have visited for multiple times as sensitive PoIs, since those places could contain more private and sensitive information [11], [19], [21].

The location requests from foreground often happen when users stop somewhere and operate the apps. Therefore, the locations collected by foreground apps only represent those places where users have shown up, but not always be the PoIs. In addition, the locations collected by foreground apps are usually discrete locations, which lack of the connection between any two of them. Differently, apps running in background frequently request to update locations according to our observations. The apps requesting location in high frequency could have a complete location trace of a user. An adversary can use the trace to identify the places where users have stayed for some time. Also, they can infer more information based on the connection between two PoIs. Generally, it would be more secure if the adversary cannot access the locations of sensitive PoIs and the locations of other PoIs as few as possible.

We adopt the idea of Spatio-Temporal algorithm to extract PoIs from location traces. In the algorithm, we regard every location as a point that contains the latitude, longitude, and time-stamp. We define three buffers buf_{Entry}, buf_{PoI}, and buf_{Exit}, which accommodate the locations possibly belonging to the PoI. Each buffer has a centroid that is calculated by averaging the latitude and longitude of all points in the buffer. The buf_{Entry} represents the area where a user starts to enter a PoI. The buf_{PoI} accommodates all the points belonging to the PoI. The second half of buf_{Entry} overlaps with the beginning of buf_{PoI}. In other words, the two buffers have an overlapped area whose size is the half of buf_{Entry}. The buf_{Exit} represents the area where a user starts to exit from the PoI. Similarly, buf_{PoI} and buf_{Exit} have an overlapped area. We fill the three buffers in sequence. For every new arrival point, we update the centroid of its buffer, which is used to calculate the distance between two buffers. When the distance between buf_{Entry} and buf_{PoI} is less than a threshold, we regard the user has entered in a PoI. When the distance between buf_{PoI} and buf_{Exit} is larger than the threshold, we regard that the user starts to exit the PoI.

Users can block the access to sensitive locations. However, an adversary is still able to infer the important information of users from the other PoIs. In this work, we consider how to prevent the adversary from inferring the movement pattern of users. We use the metric His_{bin} to test whether releasing a location will incur the leakage of user's privacy or not. While the distribution of visits can uniquely identify a user [35], [11], the connection between two PoIs could also distinguish a user from the others. This is because people may demonstrate different movement patterns when visiting a group of locations. Assume that a set of users have visited (PoI_i, PoI_j, PoI_k), some of them are used to moving from PoI_i to PoI_k, but some others may prefer to moving from PoI_j to PoI_k. Such a pattern, as a quasi-identifier [6], can be used to identify a small anonymity set. So, we take the movement pattern into account for risk detection. We create the histograms, which contain the movement pattern ($PoI_i \rightarrow PoI_j$) as the key and the number of times it happens as the value, to represent the user's profile. The first histogram is created from the complete location trace of a user as the ground truth, and the second one is generated based on the collected locations. If the two histograms match well with each other, the collected locations by the apps will indicate the users activity profile, and thus, the released locations will result in the leakage of a user's privacy

We use Pearson's Chi-square goodness of fit test [26] to decide the result of match. Assume that a user moves following a path as $P = p_1, p_2, \ldots, p_i, \ldots, p_n$, where p_i represents a PoI

of the user. We create the histogram with items $((p_i, p_j), c_{ij}))$, where (p_i, p_j) means that the user moves from p_i to p_j, and c_{ij} means the number of times the user has moved from p_i to p_j. In our context, the statistic test is shown in Formula 1,

$$\chi^2 = \sum_{i=1}^{|Pa|} \frac{c_{ij} - e_{ij}}{e_{ij}} \tag{1}$$

where e_{ij} as the expected number represents the number of times the movement pattern $(i \rightarrow j)$ happens in the user's profile, and c_{ij} represents the observed number from the collected data. The statistics converge with the degree $|Pa| - 1$ when the null hypothesis holds. We test the lower tail of Chi-square statistic, where the null hypothesis can be rejected, if the returned p-value is smaller than a threshold. In such a case, it means that the histogram from the collected data does not match that of the profile. We set $His_{bin} = 0$. Otherwise, the two histograms are similar, which means releasing the location is unsecure. We set $His_{bin} = 1$.

We further measure the information leakage from the adversary's perspective. The adversary may has acquired multiple users' location data, as the profiles, from various sources [18]. When the adversary newly receives data from a user, he could compare the data to the profiles for inferring the user's identity. Such information leakage can be measured by entropy [23], [8], which reflects the degree of anonymity. Assume that the adversary tries to infer the identity of a user by comparing the user's data to the profiles collected previously. Let X be the discrete random variable with the probability mass function $p_i = Pr(X = i)$, where X represents a user and p_i is the corresponding probability that the adversary knows the user could be the ith one. The probabilities are calculated according to the result of Chi-square test. Assume that the user's data matches the profiles of five users in Chi-square test. In other words, $\chi_1^2, \chi_2^2, \chi_3^2, \chi_4^2, \chi_5^2$ are larger than the threshold value in the test. The probability, of which the user could be the ith one, is calculated as the following Formula 2.

$$p_i = \frac{\chi_i^2}{\chi_1^2 + \chi_2^2 + \chi_3^2 + \chi_4^2 + \chi_5^2} \tag{2}$$

Then, the entropy after the inference attack is calculated by Formula 3.

$$H(X) = -\sum p_i \log_2(p_i) \tag{3}$$

Let H_M be the max entropy. The actual size of the anonymity set is calculated in Formula 4,

$$H(M) = \log_2(N) \tag{4}$$

where N is the number of those profiles that the adversary possesses. The information that the adversary traces out after the inference is calculated by $H(M) - H(X)$. Finally, we need to divide the difference by $H(M)$ for normalization, and the degree of anonymity is expressed in Formula 5.

$$Deg_{anonymity} = 1 - \frac{H(M) - H(X)}{H(M)} = \frac{H(X)}{H(M)} \tag{5}$$

TABLE III: Parameters set for extracting algorithm.

Set ID	1	2	3	4	5	6
Visiting time (min)	10	20	30	10	20	30
Radius (m)	50	50	50	100	100	100

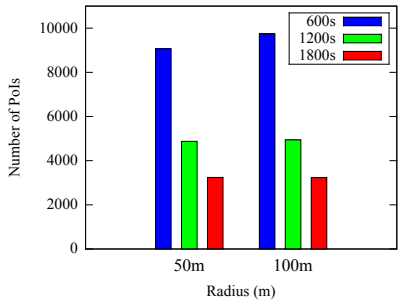

Fig. 2: PoIs extracted under different parameters

A smaller degree means a more serious information leakage. The degree equals zero if the received data matches only one profile. In such a case, the adversary knows that the data belongs to the user. On the other hand, the equal probabilities bring the adversary the max anonymity set. In such a case, the entropy equals one, and releasing the location does not expose a user's identity to the adversary. Note that the entropy is not used to detect the privacy risk for users but measure the seriousness of information leakage.

C. Measurement of the potential risks

We measure the potential risks caused by location access by using the metrics above. The measurement is conducted using the real location traces of Geolife project [37]. The dataset contains the GPS locations from 182 users. There are 17,621 trajectories covering a distance of about 1.2 million kilometers. Each entry in the dataset contains time-stamp, latitude, and longitude. About 91% of the location traces are recorded in high frequency, e.g., every 1 to 5 seconds. The dataset mainly records users' outdoor activities, including going home, shopping, sightseeing, dining, and etc. The dataset fits well for analyzing the risks caused by continuous location accesses, especially when they happen in high frequency. The interval between two consecutive locations is just one second in most cases. Thus, the original location trace is fine-grained enough to be used as the ground truth in the measurement.

First, we measure the two PoI related metrics, PoI_{total} and $PoI_{sensitive}$, according to different values of radius and visiting time. As shown in Table III, we set the radius to 50 or 100 meters, and the visiting time to 10, 20, or 30 minutes. Figure 2 shows the number of PoIs extracted under the different combinations of the parameters. Under the same radius, the amount of PoIs decrease while the visiting time increases. This is because a short visiting period can be fulfilled more easily compared to a long visiting period during the PoI extraction. Similarly, the larger the radius, the more PoIs can be extracted under the same visiting time. Overall, the impact of the radius is not as significant as that of the visiting time. We choose

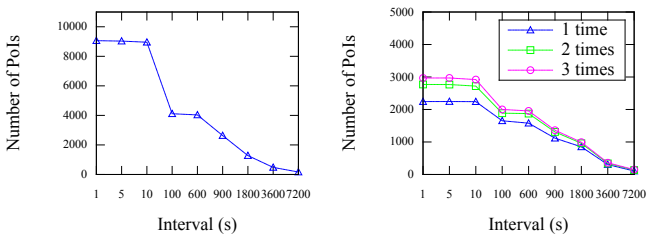

Fig. 3: (a) The number of PoI_{total} changes with the access frequency. (b) The number of PoI_{part} changes with different frequency.

the first set of parameters for our measurements, since a large number of PoIs can well describe the movement habituations of users.

Then, we show how the access frequency impacts the two metrics in Figure 3. Generally, higher frequency access allows apps to collect more complete location traces, from which we can extract more PoIs. Figure 3 (a) shows that 9,061 PoIs are extracted from the location trace collected under the frequency of one access per second. As the frequency decreases, the number of PoIs drops as well. When the interval between two location accesses is smaller than 10 seconds, there is little difference in PoI extraction. However, if the interval increases to 7,200 seconds, only around 1.8% PoIs can be extracted. And about 45.1% of apps can acquire all PoIs of users.

Similarly, the access frequency also impacts the extraction of sensitive PoIs. In the measurement, we consider those PoIs that users have visited for no more than 3 times as sensitive PoIs. Figure 3 (b) shows that the numbers of the three types of sensitive PoIs are reduced when the frequency of location access decreases. And about 45.1% of apps can trace out all sensitive PoIs. In short, a large proportion of the apps (more than 45%), which frequently update location in background, can lead to the exposure of PoIs.

In what follows, we measure the third metric His_{bin} according to the following two patterns. The first pattern ⟨region, visited times⟩ used in previous work is marked as "pattern 1", and the second one ⟨movement pattern, happen times⟩ is marked as "pattern 2". A risk is detected when the collected data passes the Chi-square test and fits the user's profile well. We set the threshold p-value to a widely used one, 0.05.

The measurement results are shown in Figure 4. Figure 4 (a) illustrates how fast the risk is detected when an app accesses location per second from the beginning of the location trace. For around 52% of users, pattern 2 helps to detect the risk when the collected locations are fewer than 10% of the profiles. However, pattern 1 only detects the risk for about 13% of users under the same condition. We conduct further measurement by collecting the random positions in the location traces. Figure 4 (b) demonstrates that pattern 2 still outperforms pattern 1. The reason is that pattern 2 describes the habituation of a user's movement, which helps quickly match the two histograms.

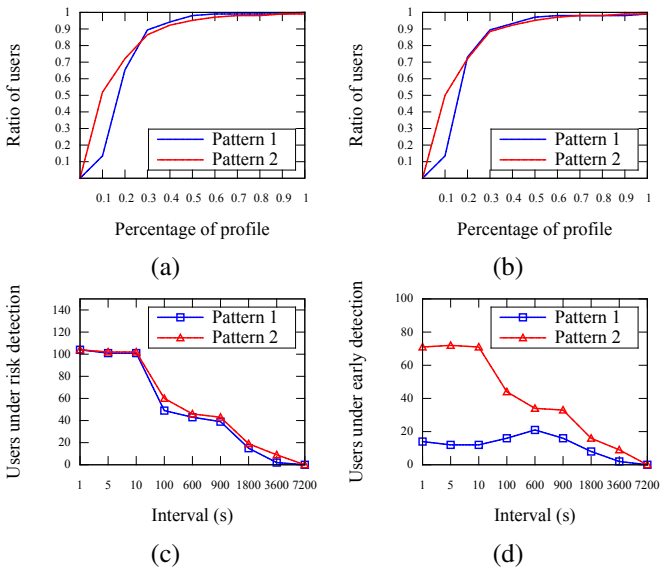

Fig. 4: (a) How many locations are needed for identification from the start point, (b) start from a random position of the traces, (c) the number of detection changes with the access frequency, and (d) the performance of the two patterns varies with the access frequency.

Specifically, pattern 2 could yield a smaller critical value and a higher degree in the Chi-square test, which contribute to the faster detection.

Moreover, we observe how access frequency impacts the risk detection. The y-axis of Figure 4 (c) represents the number of users, in whose location data the risks are detected. When one location access happens per second, both patterns detect risks for 107 users. When apps access locations in high frequency, they can collect complete location traces, which profile the users well. By contrast, if apps conduct the action in low frequency, they can only collect incomplete location traces. Therefore, the number of users drops with the decrease of access frequency. Although both patterns detect risks for the similar numbers of users, pattern 2 can detect the risk faster than pattern 1. Figure 4 (d) shows that pattern 2 achieves faster risk detection for 71 users, while pattern 1 outperforms pattern 2 only for 14 users. With the decrease of access frequency, the performance of both patterns degrades due to the incomplete location data.

At last, the measurement results of entropy are illustrated in Figure 5. A smaller entropy value represents more serious information leakage. When the frequency is one access per second, pattern 2 leads to more serious information leakage for 54 users, while pattern 1 incurs more serious privacy problem for 38 users. Similar to the measurement results of metric His_{bin}, pattern 2 outperforms pattern 1 when the access location frequency is high, and their performance will degrade with the decrease of access frequency.

In summary, pattern 2 outperforms pattern 1 in the majority of the cases when apps access location in high frequency. Meanwhile, there are some cases in which pattern 1 outper-

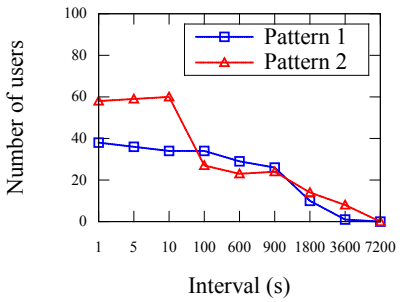

Fig. 5: Comparison of entropy.

forms pattern 2. Thus, we could combine both patterns for better results. Specifically, we can measure the privacy with both patterns and issue an alert when either of them detects the risk.

V. Conclusion

Smartphone users rarely notice a location access from background, and are unaware of the privacy risks posed by such an action. As one of the most popular mobile OSes, Android fails to handle this issue well as its permission mechanism relies on a user's decision. In this paper, we perform the first measurement study on location access from background in the Google app market. Our measurements show that many popular apps conduct location access in background. The frequency of location requests is highly related to the potential risk of privacy breach. Based on the real location traces, we demonstrate how the frequency of location requests may impact a user's privacy.

References

[1] H. Almuhimedi, F. Schaub, N. Sadeh, I. Adjerid, A. Acquisti, J. Gluck, L. F. Cranor, and Y. Agarwal. Your location has been shared 5,398 times!: A field study on mobile app privacy nudging. ACM CHI, 2015.

[2] Apktool. http://ibotpeaches.github.io/apktool.

[3] A. Bamis and A. Savvides. Lightweight extraction of frequent spatio-temporal activities from gps traces. *IEEE Real-Time Systems Symposium*, 2010.

[4] A. R. Beresford and F. Stajano. Mix zones: User privacy in location-aware services. In *IEEE Annual Conference on Pervasive Computing and Communications Workshops*, 2004.

[5] Alastair R. Beresford, Andrew Rice, Nicholas Skehin, and Ripduman Sohan. Mockdroid: Trading privacy for application functionality on smartphones. In *Proceedings of the 12th Workshop on Mobile Computing Systems and Applications*.

[6] Claudio Bettini, X. Sean Wang, and Sushil Jajodia. Protecting privacy against location-based personal identification. In *In 2nd VLDB Workshop SDM*, pages 185–199, 2005.

[7] Y. De Montjoye, C. Hidalgo, M. Verleysen, and V. Blondel. Unique in the crowd: The privacy bounds of human mobility. Sci. Rep. 3, 2013.

[8] C. Díaz, S. Seys, J. Claessens, and B. Preneel. Towards measuring anonymity. In *International Conference on Privacy Enhancing Technologies*, 2003.

[9] Dumpsys. https://source.android.com/devices/tech/debug/dum-psys.html.

[10] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Trans. Comput. Syst.*, 32(2):5:1–5:29, June 2014.

[11] Kassem Fawaz, Huan Feng, and Kang G. Shin. Anatomization and protection of mobile apps' location privacy threats. In *USENIX Security*, 2015.

[12] Kassem Fawaz and Kang G. Shin. Location privacy protection for smartphone users. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014.

[13] Adrienne Porter Felt, Erika Chin, Steve Hanna, Dawn Song, and David Wagner. Android permissions demystified. CCS '11, pages 627–638. ACM, 2011.

[14] J. Freudiger, R. Shokri, and J. Hubaux. Evaluating the privacy risk of location-based services. In *Proceedings of the 15th International Conference on Financial Cryptography and Data Security*, 2012.

[15] H. Fu, Y. Yang, N. Shingte, J. Lindqvist, and M. Gruteser. A. field study of run-time location access disclosures on android smartphones. In *USEC*, 2014.

[16] Xing Gao, Dachuan Liu, Haining Wang, and Kun Sun. Pmdroid: Permission supervision for android advertising. *IEEE SRDS*, 2015.

[17] B. Gedik and L. Liu. Location privacy in mobile systems: A personalized anonymization model. In *IEEE ICDCS*, 2005.

[18] Philippe Golle and Kurt Partridge. On the anonymity of home/work location pairs. In *International Conference on Pervasive Computing*, 2009.

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

[20] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *ACM MobiSys*, 2003.

[21] E. T. Higgins. Self-discrepancy: a theory relating self and affect. *Psychol Rev*, 94(3):319–340, July 1987.

[22] B. Hoh and M. Gruteser. Protecting location privacy through path confusion. In *SECURECOMM*, 2005.

[23] B. Hoh, M. Gruteser, H. Xiong, and A. Alrabady. Preserving privacy in gps traces via uncertainty-aware path cloaking. In *ACM CCS*, 2007.

[24] B. Hoh, M. Gruteser, Hui Xiong, and A. Alrabady. Achieving guaranteed anonymity in gps traces via uncertainty-aware path cloaking. *IEEE Transactions on Mobile Computing*, 9(8):1089–1107, Aug 2010.

[25] P. Hornyack, S. Han, J. Jung, S. Schechter, and D. Wetherall. These aren't the droids you're looking for: Retrofitting android to protect data from imperious applications. ACM CCS, 2011.

[26] David W. Hosmer and Stanley Lemesbow. Goodness of fit tests for the multiple logistic regression model. *Communications in Statistics - Theory and Methods*, 9(10):1043–1069, 1980.

[27] K. MICINSKI, P. PHELPS, and FOSTER J. S. An empirical study of location truncation on android. In *Mobile Security Technologies (MoST)*, 2013.

[28] Mohammad Nauman, Sohail Khan, and Xinwen Zhang. Apex: Extending android permission model and enforcement with user-defined runtime constraints. Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, 2010.

[29] The portal for statistics. http://www.statista.com/statistics/330695/number-of-smartphone//-users-worldwide.

[30] R. Shokri, G. Theodorakopoulos, J. Le Boudec, and J. Hubaux. Quantifying location privacy. In *IEEE Symposium on Security and Privacy*, 2011.

[31] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):571–588, October 2002.

[32] K. P. Tang, P. Keyani, J. Fogarty, and J. I. Hong. Putting people in their place: An anonymous and privacy-sensitive approach to collecting sensed data in location-based applications. In *ACM SIGCHI*, 2006.

[33] P. Wijesekera, A. Baokar, A. Hosseini, S. Egelman, D. Wagner, and K. Beznosov. Android permissions remystified: A field study on contextual integrity. In *USENIX Security*, 2015.

[34] T. Xu and Y. Cai. Feeling-based location privacy protection for location-based services. In *ACM CCS*, 2009.

[35] H. Zang and J. Bolot. Anonymization of location data does not work: A large-scale measurement study. ACM MobiCom, 2011.

[36] Y. Zheng, Y. Chen, Q. Li, X. Xie, and W. Ma. Understanding transportation modes based on gps data for web applications. *ACM Trans. Web*, 4(1), January 2010.

[37] Y. Zheng and H. Fu. *Geolife GPS trajectory dataset - User Guide*, July 2011.

[38] Yajin Zhou, Xinwen Zhang, Xuxian Jiang, and Vincent W. Freeh. Taming information-stealing smartphone applications (on android). In *Proceedings of the 4th International Conference on Trust and Trustworthy Computing*, 2011.