2019 IEEE Symposium on Security and Privacy

Demystifying Hidden Privacy Settings in Mobile Apps

Yi Chen[1,2,4*], Mingming Zha[1,4], Nan Zhang[2], Dandan Xu[1,4], Qianqian Zhao[1,4], Xuan Feng[1,4], Kan Yuan[2], Fnu Suya[3], Yuan Tian[3], Kai Chen[1,4§], XiaoFeng Wang[2§], Wei Zou[1,4]
[1]{CAS-KLONAT†, BKLONSPT‡, SKLOIS♮},Institute of Information Engineering, CAS, [2]Indiana University Bloomington,
[3]The University of Virginia,
[4]School of Cyber Security, University of Chinese Academy of Sciences
{chen481, nz3, kanyuan, xw7}@indiana.edu, {fs5xz, yuant}@virginia.edu
{zhamingming, xudandan, zhaoqianqian, fengxuan, chenkai, zouwei}@iie.ac.cn

***Abstract*—Mobile apps include privacy settings that allow their users to configure how their data should be shared. These settings, however, are often hard to locate and hard to understand by the users, even in popular apps, such as Facebook. More seriously, they are often set to share user data by default, exposing her privacy without proper consent. In this paper, we report the first systematic study on the problem, which is made possible through an in-depth analysis of user perception of the privacy settings. More specifically, we first conduct two user studies (involving nearly one thousand users) to understand privacy settings from the user's perspective, and identify these hard-to-find settings. Then we select 14 features that uniquely characterize such hidden privacy settings and utilize a novel technique called *semantics-based UI tracing* to extract them from a given app. On top of these features, a classifier is trained to automatically discover the hidden privacy settings, which together with other innovations, has been implemented into a tool called `Hound`. Over our labeled data set, the tool achieves an accuracy of 93.54%. Further running it on 100,000 latest apps from both Google Play and third-party markets, we find that over a third (36.29%) of the privacy settings identified from these apps are "hidden". Looking into these settings, we observe that they become hard to discover and hard to understand primarily due to the problematic categorization on the apps' user interfaces and/or confusing descriptions. Further importantly, though more privacy options have been offered to the user over time, also discovered is the persistence of their usability issue, which becomes even more serious, e.g., originally easy-to-find settings now harder to locate. And among all such hidden privacy settings, 82.16% are set to leak user privacy by default. We provide suggestions for improving the usability of these privacy settings at the end of our study.**

I. Introduction

Today many mobile applications (*app* for short) are designed to utilize user information for better services. For this purpose, they often seek the users' consents through various privacy settings: e.g., those for one to decide whether to share her location information with a social app or enable her friends to locate her via her phone number (*"let other users find me using my phone number"*). Although such privacy settings indeed provide the user with means to choose her desired levels of privacy protection, there are complaints about the difficulty in locating them from an app's user interfaces (UIs), which can become a serious privacy concern when these settings are by default configured to expose user data. In October 2016, Facebook was given the "Big Brother" award [1] as the "biggest privacy-offender of the year". Voters criticized its app's privacy settings, many of which are not only opt-in by default for collecting sensitive data from users (e.g., location, friend list, etc.), but also hiding deeply inside the UIs and difficult to find. For instance, Facebook had a privacy setting *"Nearby Friends"* that allowed the app to share user location with friends by default, as shown in Figure 1; it was placed under *Location* of *Account Settings*, which is *not* a typical place where users look for privacy settings, not to mention that the entry for *Account Settings* is in the middle of a long list with 41 various configurations, making it even less likely for the user to notice the entry. This problem is found to be pervasive in our study, affecting prominent apps like LinkedIn, Instagram, and Spotify.

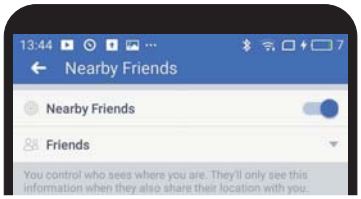

Fig. 1: Example of default privacy setting in Facebook

Understanding hidden privacy settings. In this paper, we report the first large-scale measurement study on such hidden privacy settings, which sheds new light on their pervasiveness, privacy implications and the fundamental causes of their problematic designs. For this purpose, we perform user studies to identify the unique features of the configurations hard to locate by ordinary users, and further develop an automatic analysis tool called `Hound` to recover these features from an app and detect its hidden settings.

More specifically, we conduct two user studies in our research. The first is to understand the user's perception of

*Work was done when the first author was at Indiana University Bloomington.
§Corresponding Authors
†Key Laboratory of Network Assessment Technology, CAS.
‡Beijing Key Laboratory of Network Security and Protection Technology
♮State Key Laboratory of Information Security, IIE, CAS

DOI 10.1109/SP.2019.00054

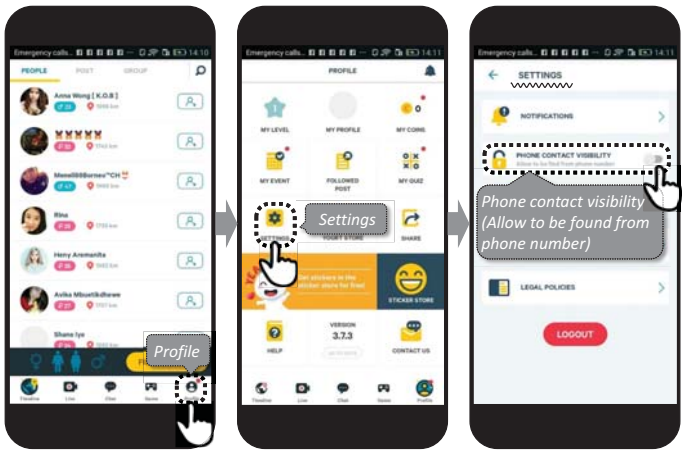

Fig. 2: Example of a UI-path

Fig. 3: Example of typical icons

Fig. 4: Example of too long text description

data exposure controlled by the privacy settings. The study shows that the overwhelming majority of the participants (83.5%) do care about at least one such privacy setting in our questionnaire. The second study is meant to find out whether these valuable privacy settings are presented to the user in the right way, which reveals that nearly half of them (47.12%) are considered difficult to find, and 9.64% cannot be located by *any* participants. From the participants' inputs, we highlight the root causes of the troubles in finding these settings and convert them into features to help detect the settings.

Detecting hidden settings. To understand the privacy implications of the hidden settings and support more usable privacy configurations, we build a new tool called `Hound` to recover privacy settings and identify those problematic ones from an app's UI. Note that finding privacy-related UI items is known to be hard [2], due to the diversity of the confidential data (e.g., friend list, music listening history), not limited to these protected by permissions. More challenging here is to detect these *hidden* ones, the settings hard to find by ordinary users, which has *never* been done before.

To address these challenges, we first utilize natural language processing (NLP) to capture privacy-related settings, through training a classifier on top of a set of feature vectors, each constructed from a setting's description (Section III). From these settings, our approach further discovers those considered to be hidden, based upon the aforementioned features identified from the user study. These features are all related to UIs, extracted from the *UI-path* that links the app's home view (i.e., initial activity) to a given privacy setting, when the user clicks on a certain UI element on each view to move the UI to the next one, until the setting is reached. An example of such a path, as illustrated in Figure 2, is $home\ view \rightarrow Profile \rightarrow Settings$. Automatic discovery of such a path through a static analysis is difficult, due to the diversity in the way that a UI transition can be triggered (e.g., intent, callback, etc.). Our solution is *semantics-based UI tracing*, a novel technique that exploits the observation that when the UI moves from one view to another, the text description of the UI element triggering the move on the former is often semantically related to the title of the latter. For example, in Figure 2, the title of the last view and the text description of its triggering UI element in the second view have the same text *"Settings"*. In this way, our approach can automatically piece together views that link the source (home view) to the target (the privacy setting). Using the features extracted from such UI-paths, Hound can effectively detect hidden privacy settings (an accuracy of 93.54%, see Section IV) with a performance of 530 seconds per app.

Findings. Running Hound on 50,000 apps from Google Play and 50,000 apps from Chinese third-party markets, we perform the first large-scale analysis of mobile apps' privacy settings. To our surprise, among all 7,058 apps containing privacy configurations, nearly half of them (47.04%) have some hidden privacy settings. Moreover, these hidden ones cover more than one-third (36.29%) of all the privacy settings discovered. Most of these hidden settings turn out to be privacy-critical: 82.16% of them by default leak out user information. Our study shows that although apparently app developers try to provide users with more privacy-related options to customize their protection, the problem of hidden privacy settings becomes even more serious, with more settings hard to discover by the user, possibly due to the developers' lack of understanding about the usability challenges from the user's perspective. Our research further reveals the developer's design pitfalls and offers suggestions for improvement.

Contributions. The contributions of the paper are summarized as follows:

• *New understanding of hidden privacy settings.* We conduct the first systematic study on the privacy settings hard to discover and understand across a large number of popular Android apps (100K). Our study brings to light the significant impacts of the problem (over a third of privacy settings hidden and most of these settings exposing user data by default) and its potential causes, which can lead to better design of these settings' UIs and enhancement of protection for user data.

• *New technique.* We develop a novel technique to automatically discover these hidden privacy settings, base upon the knowledge recovered from two user studies and a novel static analysis technique that utilizes semantics of UI elements to correlate the views triggered sequentially during the user's navigation. We demonstrate the effectiveness of the new technique, which achieves a high accuracy of 93.54%, and its efficiency that enables the large-scale study.

II. Understanding Hidden Privacy Settings

We design two user studies to understand the privacy settings from the user's perspective. One is to find out one's perception of the data exposure controlled by the privacy settings. The other is to determine whether these settings are presented to the user in an easy-to-find way. From their feedback, we further identify what makes these settings hard to find, which further helps us detect them in apps.

A. Privacy Settings in Mobile Apps

Privacy settings. To support users with personalized services, mobile apps often rely on their personal data, which may contain sensitive information. For example, many apps (e.g., LinkedIn) display to the user personalized advertisements based upon analyzing her cookies (often with sensitive information such as previously visited websites and shopping history); Facebook shares a user's friend list with others for expanding their friend circles. To get the user's consent for collecting such data and also enable her to trade protection for usability, the app developer tends to provide privacy settings to let one control how her personal information is collected, shared and used [3]. For instance, LinkedIn allows their users to disable target advertising through *"Advertising preferences (Choose whether LinkedIn can use cookies to personalize ads)"*.

To understand the privacy settings and the user data they protect, we manually analyze 200 popular apps[1] (top 100 apps in English language from the official market Google Play and top 100 apps in Chinese language from the Baidu market, a famous Chinese alternative market). These apps fall into 37 different categories as defined by Google Play (e.g., dating, finance, shopping, etc.). For each of them, we extract all the settings under the UI titled *Privacy Settings*. In this way, we collect around 600 settings and further classify them into six categories according to the data they protect, as shown in Table I, including on-device data, users' personal profile, users' social connections, users' behaviors, users' posted contents, and anti-spam settings. It is important to note that only the on-device data can be secured by system permissions such as locations and contact lists. For the data in other categories that actually constitute most (87.31%) of the privacy settings, users can only control their leakage through the settings. If they are not properly presented and managed, information will be disclosed without the user's consent.

[1]We collect the apps which have a UI titled *Privacy Settings*.

User study 1: User perspective of privacy settings. To understand whether users really care about the private data protected by the settings, we conducted an online survey through Amazon Mechanical Turk (MTurk) [4]. The survey is designed to ask users to which extent they care about the private data. Specifically, from the Facebook app, we randomly selected one privacy setting in each of the above categories, and asked the participants about their feelings if the data protected by the given settings were leaked (e.g., *"Facebook accesses your location even when you are NOT using the app. What do you feel about if this case happens?"*). A participant's response can be "Very upset", "Upset", "Neutral", "It might be okay" or "I don't care". Details of the privacy settings and the survey are presented in Appendix X-A.

The survey started in September 2018 and had lasted for two weeks. 269 Turkers participated in the survey, and 265 of them had used the Facebook app for over one month. Their median age is 29 (44.15% male and 55.85% female). 60.38% hold a Bachelor or a higher degree. After collecting all the responses, we removed 65 careless responses by checking attention questions [5]. From the survey, we find that 83.5%[2] of the participants care about (i.e., feel "Very upset" or "Upset" to) the data covered by at least one privacy setting in the questionnaire, and 61.5%[3] of them care about the data protected under more than half of the settings. Detailed responses are presented in Figure 12 (Appendix X-B). We also asked the participants to compare the data protected by the settings (provided by developers) with those guarded by system permissions [6]. 71.0%[4] of them think that the privacy-setting related data are as important as or even more important than those covered by permissions, just as we expect.

B. Hidden Privacy Settings

To understand why some privacy settings are difficult to find from the user's perspective, we conduct the second user study that involves 732 participants and lasts over 100 days. The results reveal that nearly half (47.12%) of the settings are considered as hidden, and about one-tenth (9.64%) of them are never successfully found by any participants. From the participants' feedback, we further identify six *root causes* representing almost all of their opinions about what made a privacy setting difficult to find.

User study 2: Identify hidden privacy settings. In this survey, we asked the participants to locate the 600 given privacy settings from the 200 apps and describe their experience in finding these settings. For the English apps, we recruited 405 participants from MTurk during Oct. 23th, 2017 and Jan. 18th, 2018. We installed them on an online mobile simulator Appetize.io [7] and gave the URL links to the participants for testing[5]. For the Chinese apps, we recruited Chinese participants from university

[2]79.62% if the careless responses are included.
[3]56.23% if the careless responses are included.
[4]63.94% if the careless responses are included.
[5]The mobile simulator is commonly used for testing mobile apps. The operation experience is very similar to that on real smartphones.

TABLE I: Categories and examples of privacy settings

Category	Percentage	Example	App
On-device data	12.69%	*Do not share contacts with Wicker*	Wickr
		Use your location to find people near you	Swarm
		Sync message automatically	Baiduyun
		Sync phone album	Leishiyun
Users' personal profile	14.02%	*Hide profile*	2go
		Show age on your profile	Scruff
		Only people you authorize will be able to view your profile	We Heart It
		Do you want search engines outside of Facebook to link to your profile	Facebook
Users' social connections	10.18%	*Who can see your friend list*	Facebook
		Do not show people I follow and groups I've joined	Blued
		Who can see the people, pages and lists you follow	Facebook
		Show my followers	Lofter
Users' behaviors	13.36%	*Don't show my listening history to my friends*	QQ Music
		Help accelerate migraine research with your use data	Migrain Buddy
		Choose whether LinkedIn can use cookies to personalize ads	LinkedIn
		Allow Yelp to target ads on other sites and apps based on your use of Yelp	Yelp
Users' posted content	11.69%	*Who can see your posts*	Facebook
		Show my private photos	Blued
		Who can see my diary	QQ
		Hide my articles	Rela
Anti-spam	38.06%	*Allow strangers to comment*	Zalo
		Allow friend add me without my confirmation	WeChat
		Who can send you friend request	Facebook
		Don't accept if the user has no profile icon	Airtripp

campuses for in-lab testing, mainly due to the fact that very few Turkers know Chinese. This study started on Jun. 10th, 2017 and ended on Jul. 24th, 2017, in which real mobile devices were given to the participates for doing their tasks. To minimize the bias introduced when one is unfamiliar with mobile devices, we required the participants to have at least one-year experience with the smartphone. In the experiment, each participant was asked to find five privacy settings from five apps in one questionnaire and was compensated with two dollars for Turkers and five Chinese Yuan for those attending the in-lab testing. Use the dating app YouLove as an example, which has more than five million downloads on Google Play. The participants were told *"Some social apps, such as 'YouLove', often provide settings for users to set whether to share location"*. Then we asked them to find the setting and comment on the difficulty in locating the setting ("very easy", "easy", "moderate", "difficult" and "very difficult"). If the participant considered the task to be hard, we further asked her why she thought so. Note that, to avoid unnatural user behaviors induced by the wording in the survey, our questions do not include the terms like "privacy setting" and other privacy-related wordings (which may cause users to focus more on privacy-related settings). The survey link is given in Appendix X-A.

In the end, we collected 338 completed responses from 405 participants on MTurk. Those who failed to finish the responses were mainly discouraged by the complexity of the task. Further, we removed 38 responses on MTurk and 27 responses from the in-lab test by checking whether the answers to the open-ended questions [8] make sense (e.g., They answer "Yes" to a "Why" question). In the study, 300 valid responses from MTurk (out of 338) and 300 valid responses from the in-lab participants (out of 327, with 5 valid responses for each questionnaire) were gathered before the survey collection stopped. Among all the participants, 64.66% are male and 35.34% are female, with 77.12% holding a Bachelor or higher degree.

To quantify the difficulty in locating a privacy setting, we score the responses we received. We first assign each difficulty level ("very easy", "easy", "moderate", "difficult" or "very difficult") a numeric value, with 1 being the lowest and 5 the highest. When a participant did not find the given privacy setting, we gave the setting a 5. Then for each privacy setting, we calculated its average score. If the value is above 3, we view it as a hidden setting. Among the 600 privacy settings, we find that nearly half of them (47.12%) are hidden: 50.50% of them are in English apps, and 44.33% are in Chinese apps. Among the hidden privacy settings, 9.64% have not been found by anyone; 42.83% of them have been missed by at least one participant. In our study, we calculated the Fleiss's kappa [9] for every setting across all 5 participants, and the average result is 71.93% (69.77% in English and 74.08% in Chinese), which indicates that these participants' ratings of difficulty levels for the settings are in reliable agreement.

Root causes for hidden privacy settings. Also we asked the participants to report why a setting is difficult to find. From the survey, we get 1,800 feedback on privacy settings, which shed light on the problem from the user's perspective. For example, 116 participants (reporting 502 feedback) complain that some apps do not put some privacy settings in their usual locations as other apps do: e.g.,*"I don't see it anywhere. I go to privacy setting. It's not anywhere to be found. You seriously have it there?"*. 89 participants (reporting 351 feedback) mention that it is hard for them to locate a privacy setting from a long list of different configurations: e.g., *"Too many items and texts in one screen annoys me. I don't want to read them one by one."* These feedback are analyzed in our research to identify six main causes, as shown in Table II, which cover 98.67% of all the feedback. The rest 1.33% are less common, such as *"The fonts are too small."* that is reported by only one participant. The percentage of each root cause summarized from the feedback is shown in Table II.

These six causes are all related to *UI-path*, a sequence of views that link an app's home view (i.e., initial activity) to

TABLE II: Root causes for hidden privacy settings and distributions of feedback

	No.	Causes	#(%)Feedback
Habit	1	The UI-path is uncommon.	502(27.89%)
	2	The indicator is uncommon.	153(8.50%)
	3	The UI-path doesn't have enough text descriptions of indicators.	309(17.17%)
Patience	4	The UI-path is too long.	146(8.11%)
	5	A view contains too many UI elements.	315(17.50%)
	6	The text of a privacy setting is too long.	351(19.50%)

a given privacy setting. When the user clicks on a certain UI element on a view to move the UI to the next one, these two views are connected (with direction) on the UI-path. The last view containing the privacy setting is referred to as the *key view*. For example, in Figure 2, the left screenshot is the home view, and the right one is the *key view* hosting the privacy setting *"PHONE CONTACT VISIBILITY (Allow to be found from phone number)"*. The UI-path is $home\ view \rightarrow Profile \rightarrow Settings$. Here the UI elements trigger the UI transition are called *indicators* (e.g., "Profile" and "Settings" in the first two UIs in Figure 2).

Coming back to the causes for the observed user confusion, the first three are related to user habits. Most users tend to follow the UI-path: $home\ view \rightarrow Settings \rightarrow Privacy\ Settings$. For the settings that cannot be found through this path, they could get lost (`Cause 1`). For example, we find a privacy configuration on the view titled *Messages*, which is very hard for the user to locate. Further the UI items that lead the user down the path are often characterized by certain text descriptions and icons. Once these indicators cannot be identified from such familiar signs, the user can have trouble in finding the related settings (`Cause 2`). For example, *Privacy Shortcuts* rather than *Privacy Settings* is found to make it harder for the user to get to the settings. As another example, we observe that the participants were confused when *Settings* and *Profile* are not represented by their typical icons (gear and portrait as illustrated in Figure 3). Another cause of confusion is the lack of text descriptions. We discover that participants can get lost in the presence of the indicators that do not have text and represented with unusual icons (`Cause 3`).

Also causing the problem is the user's patience. From the participants' feedback, in the presence of a long UI-path (`Cause 4`) or a view with too many UI elements (`Cause 5`), the user becomes less likely to find a privacy setting. For example, a "difficult" setting reported in our study is hidden behind a UI-path with 5 views, which can only be reached after the user clicks on 4 indicators, as shown in Figure 13 at Appendix X-B. We even found a view carrying 47 elements. Its privacy settings cannot be fully identified by all participants in our study. Also interesting is that when a setting's text description is too long (see Figure 4), most people simply ignore the setting (`Cause 6`).

III. Discover Hidden Privacy Settings

To understand the privacy implications of hidden settings, we develop an automatic tool called `Hound` and utilize it to perform a large-scale study on the usability of mobile apps' privacy configurations. Here we elaborate the design and implementation of the tool.

A. Design Overview

The high-level design of `Hound` is illustrated in Figure 5, which includes two components: *Privacy Setting Discoverer* (PSD) and *Hidden Privacy Setting Identifier* (HPSI). PSD runs Natural Language Processing (NLP) on the descriptions of an app's settings to identify those related to privacy. Then from those settings, HPSI further utilizes a classifier, built on top of the features discovered from our user study (see Section II-B), to find those considered to be hidden.

PSD. PSD is designed to extract all the settings from a given app, and then filter out those unrelated to user privacy based on its analysis of their text descriptions. More specifically, its *setting extractor* statically analyzes the app's UI layout files after disassembling it to find all its settings. The layout files are in XML whose vocabulary [10] describes different UI widgets. For example, the setting option widgets are usually represented by $< CheckBox.../ >$ [11], $< Switch.../ >$ [12] or $< ToggleButton.../ >$ [13]. Moreover, the text descriptions for the settings are in the widgets' $android:text$ attributes or in those of other widgets nearby. Through checking widget types and their text description, all the settings can be statically discovered from the layout files[6].

For each setting discovered, PSD uses NLP to determine whether it is privacy-related. For this purpose, it first runs *vectorizer* to build a vocabulary with all 1-gram and 2-gram terms in the description of each setting. Then it inspects the description and transform it into a numeric feature vector based on tf-idf (frequency-inverse document frequency), a standard term-weighting scheme that measures the importance of a word in the document. In this way, we can model each setting description with a high-dimensional (the size of the vocabulary) feature vector. The feature vector is then used by the *privacy classifier* to determine whether the setting is privacy-related. The implementation of the PSD component is elaborated in Section III-D.

HPSI. Most challenging here is how to detect *hidden* privacy settings, those considered to be hard to find by the ordinary user, which has never been done before. To address this challenge, HPSI leverages the feedbacks obtained from our user study (Section II-B) to train a classifier for predicting whether a setting has the usability issue. More specifically, for each privacy setting reported by PSD, HPSI extracts from it a set of features and decides whether it is hidden by running a classifier (*hiddenness classifier*) on the features.

Altogether, 14 features are used to characterize the hidden settings, which are numeric values assigned based upon the aforementioned six causes summarized over 1,800 feedbacks

[6]Some apps dynamically generate widgets and draw them in a view, which is out of the scope of this paper. To capture them, one could analyze the code for creating widgets in the apps.

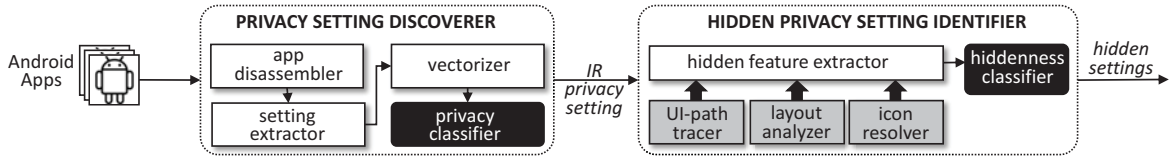

Fig 5: Architecture of Hound.

received in the second user study (Section II-B). Each feature is designed to capture one aspect of the UI design that makes a privacy setting difficult to find. They are elaborated in Section III-B.

Extracting these features from an app's code turns out to be yet another challenge, due to their dependence on the UI-path is nontrivial to construct statically. Our solution is *semantics-based UI tracing* that connects the views involved in a UI transition based upon the semantics of their content, particularly between the text description of an UI element and the title of the view that the element invokes once it is triggered. This allows us to build an accurate UI-path automatically and further extract features from the path, as elaborated in Section III-C.

B. Features of Hidden Privacy Settings

As mentioned earlier, we have summarized six causes that render a privacy setting difficult to locate by the user (Section II-B). They are further quantified into 14 features as shown in Table X (in Appendix X-B) to characterize a setting, which could further be utilized to determine hidden settings. Below we detail the features.

`Cause 1`: Users could get lost when a privacy setting that they are looking for does not follow a typical UI-path (e.g., $home\ view \rightarrow Settings \rightarrow Privacy\ Settings$). Thus, we measure the similarities between a given setting's UI-path and a set of typical privacy-setting UI-paths, as identified by the participants of our user study from 200 popular apps, through calculating the Jaccard indices [14] between the sets of nodes on the two given UI-paths u_1 and u_2. The maximum value of such similarity scores is chosen as the first feature `F1`. Also, since most users look for a privacy setting on $Privacy\ Settings$ or $Settings$, we utilize `F2` and `F3` to describe whether a given privacy setting is on the two views, respectively.

`Cause 2`: Developers sometimes use uncommon text or icons as indicators on UI-paths, which may not help users find a given privacy setting. We design three features (`F4` $\sim$ `F6`) to characterize these situations. For the uncommon text indicators, we measure the similarities between each text indicator on a given UI-path and a set of most common indicator terms (e.g., *Me*, *Settings*, *Privacy Settings*, etc.), which have been collected manually from 200 popular apps. The similarity here is calculated as the maximal Jaccard Index between the words in an indicator text and those in the common term set. The average similarity of all the indicators along a UI-path becomes a feature `F4`. Secondly, we utilize the *privacy classifier* to determine whether an indicator is related to privacy. The total number of those unrelated to privacy is used as the feature `F5`. Finally, from the feedbacks of our user study, people only consider gears (standing for *Settings*) and portraits (standing for *Profiles*) as icon indicators (in Figure 3) for configurations. If developers use other icons on a UI-path, the users probably will not click on them to seek privacy settings. So we count the number of the icons down a UI-path that are neither gears or portraits as the sixth feature `F6`. Since the developers may come up with an icon in very diverse ways, we cannot use image comparison for determining whether an icon is similar to a commonly used gear icon. Our solution is to leverage "Best Guess" in Google, which could give the semantic meaning of the icon, for accurate comparison. An example of such a "Best Guess" is shown in Figure 6.

`Cause 3`: Too many icons on a UI-path without text descriptions could confuse users. We count the number of icons in a UI-path as a feature `F7`.

`Cause 4`: A long UI-path may let users lose patience to find a privacy setting. So we count the number of views on a UI-path as another feature `F8`.

`Cause 5`: Too many UI elements on a view could also cause one to lose patience. After carefully analyzing the users' feedbacks, we use five features to describe this cause. Since the key view (the one displaying the privacy setting) is important for users to locate the privacy setting they are looking for, we count the number of the UI elements on the key view as the feature `F9`. Sometimes, users complain that when privacy settings and other settings are mixed together, it becomes hard for them to single out the privacy ones. So we calculate the percentage of the privacy settings on the key view as `F10`. Besides the key view, a similar problem also happens to other views on a UI-path. So we count the number of the settings for each of the view on the UI-path and choose the maximum one as `F11`. Normally, users browse a view from the top to the bottom and thus the settings located close to the bottom may have a higher possibility to be missed. So, we use the privacy setting's position from the top of the key view as `F12`. Similarly, across other views on the UI-path, we use the average of the individual indicator's positions on them as `F13`.

`Cause 6`: A very long text description of a privacy setting may also cause the users to lose patience. So we count the number of words in the descriptions as `F14`.

With all these features, we are able to quantify the difficulty of locating a privacy setting, and identify the hidden ones from users' perspective. However, to automate this process at a large scale, Hound has to build UI-paths for given privacy settings in an app, which turns out to be more challenging than we thought. Particularly, to the best of our knowledge, no prior approach can be directly applied to precisely extract UI-paths

Fig. 6: Example of Google "Best Guess"

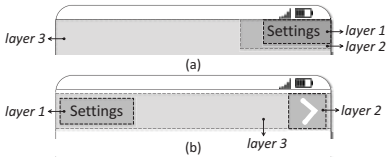

Fig. 7: Example of overlapped widgets

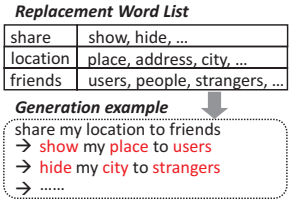

Fig. 8: Example of synthetic sampling

from an app. Our solution to this problem is *semantics-based UI tracing*.

C. Semantics-based UI Tracing

As mentioned earlier, Hound has to recover UI-paths before quantifying the features along the path. A UI-path connects an app's views using indicators, which ends with the given privacy setting. To generate a UI-path, one may think of dynamic analysis (i.e., enumerating all the UI-paths by triggering various events in the app). However, this approach is time-consuming and can only find a limited number of UI-paths. Our idea is to statically find views and indicators in the app, and then to link them together for building UI-paths. This is also very challenging due to the complexity of the layouts and the limitation of static analysis, as elaborated below.

Challenges. The first challenge is that the indicator clicked by users may not be correctly identified by simply analyzing app code. From the users' perspective, they focus on the texts displayed on a view, and regard the widget [15] holding the texts as the indicators. However, code analysis may view the widget that handles the user's click as the indicator, that is not the one seen by the user. For example, in Figure 7 (a), there are several overlapped widgets. The widget displaying the text (in layer 1) is not the one (on layer 2) handling users' click. Another example is shown in Figure 7 (b). The widget with the text (layer 1) and an indicating arrow on the right side (in layer 2) are both on the same widget (in layer 3). Users click the arrow on layer 2 because of seeing the text on layer 1, but in fact, layer 3 handles the click. Similar situations commonly exist in today's apps due to the gap between users' perceptions and code implementations. We analyzed 100 privacy settings in the top 100 apps from Google Play and found 89% (89/100) of them have this problem, which makes it hard for static analysis to identify indicators from the users' perspective.

The second challenge is the dynamic features of Java, which are hard to handle by static analysis. For example, *inheritance* is an important Java feature where the child class can extend the properties (methods and fields) of its parent class. This feature is widely used by *Activity* [16] which is the main implementation of a view in Android app development. Child *Activities* can inherit basic properties from their parent class, and implement their own properties. Using inheritance, when a developer wants to call a function in a subclass, what he does is to call the corresponding function in the superclass, without indicating the subclass to reduce the workload of programming. Java Runtime Environment [17] will decide the correct subclass according to dynamic running environments. Therefore, it is very hard for static analysis to know the correct child class for capturing the connection between indicators and views.

Our idea. To address the two challenges, we design a novel semantics-based approach to extract UI-paths, avoiding heavyweight static analysis (e.g., traditional data flow and control flow analysis). Our idea is based on an observation that the title of a view has a semantic connection with the indicator that the user clicks. For example, in Figure 2, the title of the third view "Settings" matches the text of the second indicator text (also "Settings"). This observation actually follows Google's guidelines [18] for app design: developers should let users be aware of the navigation positions in the app. We analyze 300 UI-paths in the top 100 apps from Google Play, only two views in the two UI-paths do not follow the observation (which actually may not be a good design). Hence, the connection between the title of a view and its indicator is exactly the hint for us to construct UI-paths.

Based on the observation, our idea is to leverage the semantics of indicators and views to build the connections between them, and further to construct UI-paths. Basically, Hound first extracts all the indicators and views including their semantics from an app. Then it connects an indicator and a view if they have similar semantics. By tracing from a privacy setting (i.e., the end of a UI-path) to the home view (i.e., initial activity), Hound is able to build the full UI-path step by step. In this way, we can avoid the static code analysis, and naturally address the challenges above. We detail how to extract the semantics and how to build connections as follows.

Semantics of indicators and views. There are two types of indicators: texts and icons. Both of them have semantics that guides users in the process of UI navigation. For text indicators, the texts themselves can provide enough semantics. For icon indicators, our idea is to leverage Google's "Best Guess" that we use for obtaining the semantic meaning of an icon previously (see Section III-B). By uploading an icon to Google, the semantics of the icon could be returned in the form of texts automatically. Besides using Google, we also take a look at the resource name that refers to an icon in code which may contain semantics. In this way, Hound is able to extract the semantics of all the text and icon indicators in a given app.

Regarding views, most developers choose meaningful titles to give users clear guidance on views' contents. Thus, the text of the title can be used for a view's semantics. To extract

TABLE III: Semantic sources for title determine

	Source	Example
layout	string name	android:text='@string/setting_title'
	widget id name	android:id='@id/title'
	style name	android:style='@style/center_title'
smali	method name	setTitle, initTitle, setCaption()
	class name	titleBar, headBar
	parameter/field type	progress_title, actionBar_head
	annotation	.annotation runtime AcitivtyCenterTitleRes

the titles, considering developers usually use the standard Android API "setTitle" for assigning titles to views, Hound can directly extract them from the API's parameters in app code. However, developers sometimes build their own title layout style without using the API. How to determine the text for a view's title becomes non-trivial. Our idea is based on an observation that developers would name resources, classes and methods with certain semantics to help themselves develop apps. Such semantics is a good hint for us to identify a view's title. For example, when a text's id is *"@id/title"* in a view layout file, we believe that the text has a high possibility to be the title of the view. We summarize the semantic sources we use to determine a title in Table III. In the worst case, even if we fail to find the title text using the above approach, the name of a view's layout file and its classname may also contain semantics. For example, the classname of a view for privacy setting is often named *PrivacySettingsActivity*, and the layout file is usually named *privacy_settings.layout*, which give semantics for establish the connection.

Build the connection. Hound connects the indicators and views with similar semantics and starts to build a UI-path backward from privacy settings. The basic idea is, given a privacy setting, Hound first locates the view which contains the setting. Then it searches for the indicator which has the most similar semantics as the view and puts both of them onto the UI-path. Later, Hound continues this process until the home view is reached. In particular, to compare the semantics of two texts, we define a similar semantics degree $d(text_1, text_2) = Jaccard(W_{text_1}, W_{text_2})$, W_{text_1} and W_{text_2} are the word sets of the $text_1$ and $text_2$ respectively. For example, when Hound calculates the degree for the view titled "Privacy Settings Activity" and an indicator named "Privacy Settings", W_{text_1} is {"privacy", "settings", "activity"}, W_{text_2} is {"privacy", "settings"}. So the similar semantics degree is 0.67. In this way, a UI-path can be constructed.

D. Implementation

PSD. PSD includes three modules: (1) *App disassembler* is implemented using APKTOOL 2.3.3 [19], which reverses Android apps to IR code (i.e., smali code) for further analysis. (2) *Setting extractor* utilizes UIPicker [2] to extract all the settings and their text explanations in an app. (3) *Vectorizer* and *privacy classifier* are developed using a machine-learning toolkit called scikit-learn [20], to transform setting texts into numeric vectors and train the classifier.

To train the *privacy classifier*, we gathered 12,208 text items for settings and indicators from the 200 apps (including 100 English apps and 100 Chinese apps, as mentioned in Section II-A). Then, based on Table I, we manually labeled them as privacy-related or not. On first sight, it seems that the data could be directly used in training. However, the number of privacy-related texts is too small (only 670), while there are 11,538 non-privacy texts. The classifier trained from these data will be prone to overfitting. Therefore, we extend the number of privacy-related texts. Since manually analyzing more apps is time-consuming, our idea is to generate synthetic samples based on existing examples [21], which is commonly used for learning from imbalanced data. Particularly, for the texts of a privacy setting, we first identify verbs and nouns through Part-of-speech Tagging [22] of NLP, and change them into their synonyms and antonyms to generate new texts. For example, for the texts *"share my location to friends"*, Hound recognizes the nouns *location* and *friends*, and the verb *share*. And then Hound replaces them with other words and generates new texts such as *"hide my city to strangers"* (see Figure 8). In this way, we get 6,700 unique privacy-related texts from the original 670 privacy-related texts.

With the dataset, we tried several machine learning models to build the classifier, including decision trees [23], random forests [24] and SVM [25] with linear, poly and RBF kernels. To achieve better performance, we utilize Optunity 1.1.1 [26], a library containing various optimizers for hyperparameter tuning, to evaluate each classifier with different parameters for 10,000 times. The results show that SVM with linear kernel performs best with parameter C=2. We used 5-fold cross-validation, this model achieves a precision of 96.64%, a recall of 97.94% and an accuracy of 97.91%.

HPSI. HPSI is composed of two modules: (1) *Hidden feature extractor* is supported by three techniques. *UI-path tracer* extracts UI-paths using the *semantics-based UI-path tracing*. *Layout analyzer* uses techniques from UIPicker [2] to analyze the layout of each views. *Icon resolver* leverages "Best Guess" of Google to get the meaning of icons in UI-paths. (2) *Hiddenness classifier* is developed using the toolkit scikit-learn [20] again.

The training data for the *hiddenness classifier* is labeled by participants in the second human subject study (see Section II-B), which allows us to detect hidden privacy setting from users' perspective. In the study, they labeled 283 hidden privacy settings and 317 easy-to-find privacy settings. Then similar to train the *privacy classifier*, we use different models and choose the one with the best performance as the final model. From our experiments (5-fold cross-validation), we use SVM with RBF kernel (using the parameters: *C=3 and logGamma=-4*). Our model achieves a precision of 93.01%, a recall of 90.53% and an accuracy of 94.29%. Note that, even though the training data come from the top 200 popular apps, the evaluation of HPSI (see Section IV) demonstrates the effectiveness of the classifier on randomly selected apps and therefore suggests that our training set (the 600 privacy settings) is representative.

TABLE IV: App collection

App Source		Count
GooglePlay		50,000
Third-Party Markets	360	17,414
	Baidu	6,118
	Xiaomi	23,826
	Huawei	908
	Tecent	1,734

IV. Evaluation

We evaluate the effectiveness and performance of Hound as follows.

Setting. We crawled real-world apps from various app markets in 2017 and get 100,000 apps after removing duplicated ones according to their MD5 checksums. Among the apps, 50,000 apps are from Google Play [27] and others are from third-party markets in China. 37 categories are included (e.g., social, business, dating, etc). Details of the apps are shown in Table IV. Our server to statically analyze these apps has 80 cores with 2.2GHz CPU, 256GB memory and 70TB hard drives.

Effectiveness. The effectiveness of Hound depends on the accuracy of PSD and HPSI. Thus, first we evaluate the two components respectively and then calculate the accuracy of the whole system.

First, we evaluate how accurate PSD is to identify the privacy settings among all the settings in an app. We should have to manually check all the texts of settings to find the privacy-related ones referring to Table I and compare them with PSD's results. However, since it is pretty time-consuming to go over all apps with manual work, we randomly selected 100 apps for evaluation, including 50 apps from Google Play and 50 apps from Chinese third-party markets. In the 100 apps, PSD discovers 470 privacy settings in 7,891 settings, including 454 real privacy settings and 16 wrongly identified ones, while we find 477 privacy settings manually. Thus, PSD's precision is 96.60% (454/470), recall is 95.18% (454/477) and accuracy is 99.51% ((454+7,398)/7,891).

Secondly, we evaluate how accurate the HPSI is to identify hidden ones among all the privacy settings discovered by PSD. We ran an in-lab user study using a subset of the questionnaire for the user study 2 (see Section II), including the questions about whether settings are easy to find, not why they are easy/hard to find, to get the manual labels from the participants. Then we compared them with the prediction results of HPSI to measure its effectiveness. Specifically, we randomly chose 200 privacy settings (100 hidden ones and 100 easy-to-find ones identified by HPSI) from 200 randomly selected apps (one setting per each app, 100 English apps, and 100 Chinese apps). Then we recruited 100 participants in the U.S. for English apps and 100 participants in China for Chinese apps. Similar to the user study 2, each participant was asked to find five given settings from five different apps in one questionnaire. Then they reported the difficulty level (from very easy to very difficult) for locating a setting. In the end, we compared the users' average rating for locating a privacy setting with the prediction result produced by HPSI. Details are shown in Table V. Among the 200 privacy settings classified by HPSI (100 hidden and 100 easy-to-find ones), the participants labeled 102 as hidden and 98 as easy-to-find. After comparing the settings one by one, we conclude that 188 (95+93) privacy settings are correctly classified by HPSI, and 12 hidden privacy settings are wrongly labeled. Hence, HPSI achieves an accuracy of 94.00% (188/200). Notably, for hidden privacy settings, it achieves a precision of 95.00% (95/100), a recall of 93.13% (95/102) and an F1 score of 0.9412.

TABLE V: Evaluation results of HPSI

		Participants		
		hidden	easy-to-find	total
HPSI	hidden	95	5	100
	easy-to-find	7	93	100
	total	102	98	200

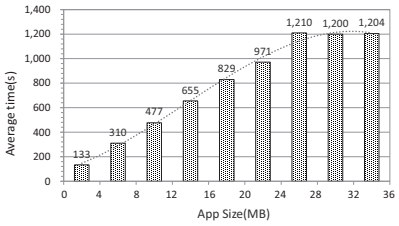

Fig. 9: Time cost distribution

For the incorrectly classified ones, we look into them and find that their average scores are very close to three, which means the privacy settings may be on the boundary. Further, we communicate with participants about the 12 settings why they view them as hidden or easy-to-find ones to figure out the reasons for the wrong classification of HPSI. For the 7 false-negative cases, HPSI identifies the icons as gears for settings and texts as privacy settings, but participants say that the icon sizes and the text font are too small for them to identify, or the color of icons (e.g., a light green gear) is less noticeable. Thus, participants miss them while our HPSI discovers them. For the 5 false-positive ones, 2 is that the HPSI fails to identify the icon for setting, but participants think it is expressive for understanding; the other 3 cases are that participants think the indicators have enough semantic information for leading to privacy settings, but HPSI thinks the indicator texts are unrelated to privacy. For this situation, in the future, we can improve the accuracy by adding the semantic relevance between indicators and privacy settings as a new feature, even though the indicator text is not related to privacy.

Therefore, while considering PSD and HPSI together, Hound can achieve an accuracy of 93.54% (99.51% * 94.00%) for hidden privacy settings identification.

Performance. Hound has analyzed over 100,000 apps. Each app costs about 530 seconds on average, including app dissembling, setting extracting, privacy discovering and hidden setting identifying. The median time of app analysis is 378 seconds, ranging from 15 seconds to 1798 seconds. It takes a longer time to analyze apps with larger code size. But from Figure 9, when the size of an app reaches 24 MB, the average time does not increase any more. We guess that it is because Hound analyzes an app from the views.

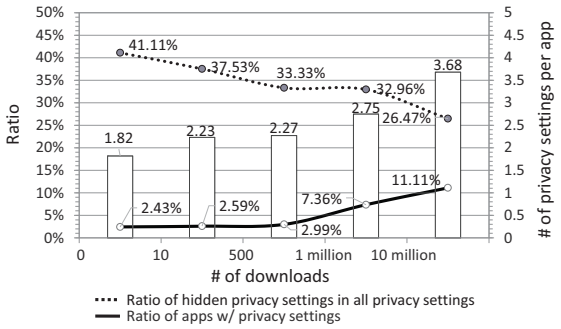

Fig. 10: Hidden privacy settings on downloads

V. Measurement

With the real feedback from two human subject studies lasting for several months and our scalable tool Hound, we are able to look into the problem of privacy settings in 100,000 popular apps and gain unprecedented understandings of them from users' perspective. Our results not only show the pervasiveness of hidden privacy settings but also disclose the behind reasons why privacy settings are viewed as hidden. Especially, through the evolution of privacy settings, we show how privacy settings are changed to hidden even if developers may actually have opposite willing. Further, we reveal that a large number of settings are set to leak user privacy by default.

A. Landscape

As shown in Table VI, 7,058 out of 100,000 apps have privacy settings. To our surprise, Hound finds that nearly half of them (47.04% = 3,320/7,058) have some hidden privacy settings. Moreover, these hidden ones cover more than one-third (36.29% = 6,723/18,526) of all the privacy settings discovered. On average, each app contains 2.03 (6,723/3,320) hidden privacy settings. For the apps with a lot of downloads, the problem of hidden privacy settings looks less severe, as shown in Figure 10. Given that the popular apps actually contain more privacy settings than those less popular, an educated guess could be that the developers of the popular apps might pay more attention to the usability of privacy settings and therefore are more likely to make these settings explicit.

We compare the number of privacy settings between English apps from Google Play and Chinese ones from the Baidu market. From Table VI, we can see that English apps have more privacy settings than Chinese apps. Specifically, 9.45% English apps contain privacy settings while only 4.67% Chinese apps do. This leads to our speculation that the developers of English apps might attach more importance to privacy. Further, among the 4,724 English apps containing privacy settings, 45.15% (2,133/4,724) include hidden ones. For Chinese apps, the ratio turns out to be similar (50.86%=1,187/2,334). Also, about one-third of all privacy settings in both English (38.49%) and Chinese apps (30.99%) are hard to find, which indicates that hidden privacy setting can be a pervasive issue for both English and Chinese apps.

B. Why Does a Privacy Setting Become Hidden?

After manually analyzing each one of the detected 1,132 hidden privacy settings from 500 randomly sampled apps, we find that there are two main fundamental causes of their problematic designs for developers. One is that developers make problematic categorization and place a privacy setting into a view unfamiliar to users though they think the place is easy for users to find. The other one is that developers use confusing descriptions on privacy settings or indicators in the way they understand but not so for users. We describe them as follows.

Problematic categorization. As mentioned previously, users expect privacy settings are in the view titled *Privacy Settings*. However, developers may place them in other views. From our evaluation, we find that 790 apps do not even have the view of *Privacy Settings* for placing their 1,247 privacy settings. 857 apps disseminate their 1,327 privacy settings in other views other than *Privacy Settings* which actually exists in the apps. We find three places where developers put hidden privacy settings.

Firstly, developers place a privacy setting to the view related to its functionality. For example, a famous travel app called Baidu Map has a functionality called *"Track"* to record all previous tracks that a user navigated before, for letting the navigation better match users' expectation. To protect users' privacy, several settings are designed to control the tracks collection, such as *"allow to record your driving/walking navigation tracks"*. The UI-path of this privacy setting is $Me \rightarrow Track \rightarrow TrackSettings \rightarrow allow\ to\ record\ your\ driving/walking\ navigation\ tracks$. Developers expect users could find the setting when using the functionality. However, from our study, it is pretty difficult for a user to locate the setting.

Secondly, developers place a privacy setting to a view whose title has similar semantics with the privacy setting. For example, a privacy setting *"Who can send me private messages"* is placed under a view titled *Message Settings* in two popular apps (i.e., a radio app named Lanrentingshu with more than 80 million downloads and a social app named Wangliao with more than 2 million downloads). Also put into *Message Settings* are anti-spam privacy settings. Even though the setting and view's title have the same keyword "message", users in our experiments never try to locate the privacy setting from this view.

Thirdly, developers put privacy settings in the view of general settings. We find 293 apps put their privacy settings in *Account* or *Notification*, and 116 apps put privacy settings into the view titled *Edit Profile*. Such design lets users get lost when they have to find the setting. Some lazy developers even mix privacy settings with others and put all of them into the *Settings* view, which results in too many settings in a view for users to locate. Generally, once there are more than ten settings in a view, users have to swipe the screen to check them all. From our evaluation, we find that 5.89% (396/6,723) hidden privacy settings are mixed with non-privacy settings, and they are located in the middle of a view with more than ten settings.

TABLE VI: Privacy settings and hidden privacy settings identified by Hound

	Number of apps			Number of settings	
	Total	**w/ privacy settings**	**w/ hidden privacy settings**	**privacy settings**	**hidden privacy settings**
Google Play	50,000	4,724(9.45%)	2,133(45.15%)	13,101	5,042(38.49%)
Third-party markets	50,000	2,334(4.67%)	1,187(50.86%)	5,425	1,681(30.99%)
Total	100,000	7,058(7.06%)	3,320(47.04%)	18,526	6,723(36.29%)

Fig. 11: Example of strange icons for settings

Taking the famous social app Instagram as an example, users have to locate the privacy setting *"Allow accounts you follow and anyone you message to see when you were last active on Instagram apps"* after swiping the screen to the 26th setting in the view which contains 38 settings and 36 of them are non-privacy.

Confusing description. Developers should describe an indicator or a setting in the way that users get used to. However, in our evaluation, we find 237 apps having hidden settings contain 271 uncommonly used descriptions which mislead users for locating a privacy setting. Typically, *Profile* and *Settings* are two indicators that almost always appear in a UI-path to privacy settings. In our evaluation, we find 58 apps use their synonyms (e.g., use "Privacy Shortcuts" instead of "Privacy Settings"). For users who get used to the most common terms, even replacing "Privacy Settings" with "Privacy Controls" will let them feel confusing. The situations are even diverse for icon indicators. We find 47 uncommonly used icons in our evaluation. Figure 11 shows some examples for *Settings*. The five icons, especially the last one (in a Chinese popular social app with more than 100 million downloads), are far from the meaning of "setting".

Another problem is that an indicator fails to let users infer the privacy settings it guides. For example, in a famous social app for job hunting named Maimai, an indicator with text *"who can see me"* guides to the privacy settings *"whether allow search engine find me"* and *"whether create a Hudon wiki for me automatically"*. Most users in our evaluation cannot infer the two privacy settings when they see the indicator. Such cases happen to 74 privacy settings in 39 apps. Also, too long texts to describe an indicator or a setting also let users lose patience. From our evaluation, we find that the texts to describe privacy settings in English apps usually range from 1 to 173 words, which is much more than that in Chinese apps (usually with 2 to 15 words). Among them, 70 English hidden settings have more than 30 words, and 24 settings have more than 50 words.

C. Evolution

To understand the problem of hidden privacy settings over time, for the 7,058 apps which have privacy settings (collected in the year of 2017), we downloaded their latest versions (in 2018). Finally, after removing those with no update, 5,485 new versions were successfully collected. For each app pair, we compare their privacy settings. Interestingly, on one hand, we find 3,110 new privacy settings appear and 1,047 privacy settings disappear. 36 apps add a new view named *Privacy Settings* to organize the privacy settings which scattered in their previous versions. On the other hand, the number of hidden privacy settings also increase. For the 5,485 app pairs, the number of apps with hidden privacy settings increases from 2,564 (46.75%) to 2,758 (50.28%). Also the number of hidden privacy settings increases from 5,212 (36.81%) to 6,847 (42.21%). More detailed is shown in Table VII. Apparently, the developers are working on privacy-related options and provide more privacy settings for users. In the meanwhile, however, their current designs still face the usability challenges. Below we dive into the details and show how privacy settings change to hidden, or vice versa.

TABLE VII: Evolution from 2017 to 2018

	year	# of apps		# of settings	
		w/ privacy settings	w/ hidden privacy settings	privacy settings	hidden privacy settings
en	2017	3,671	1,665(45.36%)	10,186	3,991(39.18%)
	2018	3,671	1,805(49.17%)	11,923	5,469(45.87%)
zh	2017	1,814	899(49.56%)	3,972	1,221(30.74%)
	2018	1,814	953(52.54%)	4,298	1,378(32.06%)
Total	2017	5,485	2,564(46.75%)	14,158	5,212(36.81%)
	2018	5,485	2,758(50.28%)	16,221	6,847(42.21%)

Easy-to-find→Hidden. Unfortunately, 226 privacy settings in 158 apps change to hidden even if they are easy-to-find in previous versions. After carefully looking into the settings, we find the causes are the same as those in Table II. When an app has more functionalities in its newer version, developers may not appropriately handle the newly added privacy settings, especially when developers' main purpose is the usability of apps. It is hard for them to balance between supporting users with enough control on privacy and keeping the app to be simple and easy to use. Further, when developers are under pressure to release a new version, they might do not have enough time to design the privacy setting. So they are rush to put privacy settings into unfamiliar places or use unfamiliar descriptions without carefully thinking. For example, Photobucket (a popular photography app with 10 million downloads in Google Play) has a privacy setting *"Share GIFs with Photobucket"*. The developers move the setting from the view titled *Settings* into a new view titled *Upload Settings* in the new version 3.3.8, which makes it difficult for users to locate the setting.

Hidden→Easy-to-find. We are also happy to find that 192 privacy settings in 132 apps become easy-to-find. Some developers gather scatted privacy settings to a view and use an indicator in the *Privacy Settings* view for guidance. For example, recall that Hound finds a hidden privacy setting in Baidu map (see Section V-B). In its new version 10.1.0 published in Feb. 2th, 2018, developers put the setting into a newly created view titled *Track Setting* and add an indicator

TABLE VIII: Privacy leakage in different languages

	English	Chinese	Total
# of privacy settings	329	270	599
# of privacy settings that leak privacy by default	260 (79.03%)	209 (77.41%)	469 (78.30%)
# of hidden privacy settings	110	75	185
# of hidden privacy settings that leak privacy by default	93 (84.55%)	59 (78.67%)	152 (82.16%)

TABLE IX: Privacy leakage in different categories

Category	Privacy settings		Hidden privacy settings	
	Total	Opt-in to leak	Total	Opt-in to leak
On-device data	76	52 (68.42%)	22	16 (72.73%)
Users' personal profile	84	69 (82.14%)	50	44 (88.00%)
Users' social connections	61	61 (100.00%)	20	20 (100.00%)
Users' behaviors	80	64 (80.00%)	28	22 (78.57%)
Users' posted content	70	46 (65.71%)	27	19 (70.37%)
Anti-spam	228	177 (77.63%)	38	31 (81.58%)

pointing to the view under the *Privacy Settings* view, which is easy for users to find. Another example is BMW Motorrad Connected (an app for BMW bike) which has a privacy setting *"Help us improve the range of products from BMW Motorrad"*. The setting lets users control whether or not to allow the app to collect user location, trace and other usage data in the background. In the old version (1.3), it was put in the view titled *Legal notices* and the UI-path does not even include the *Settings* view. In the new version (1.4.2), developers move the privacy setting into the view titled *Data privacy* which is much easier to locate. Similar changes happen on popular apps (e.g., KingsChat with 500,000+ downloads and Web Browser with 10 million downloads). We find that among the 132 apps, 98 have been downloaded more than 100 thousand times and 54 installed over 1 million times. Therefore, it appears to us that the developers of popular apps attach more importance to the designs of privacy settings and are likely making more efforts to improve their usability.

D. Privacy Leakage by Default

The problem will be even more severe when hidden privacy settings are set to leak users' private information by default. Taking Facebook as an example, 22 out of 34 privacy settings are set to leak privacy by default, among which 12 are hidden. To further understand this problem, we randomly selected 100 English apps from Google Play and 100 Chinese apps from Baidu market, and checked 599 privacy settings manually. The results are astonishing: 469 (78.30%) privacy settings are set to leak privacy by default (79.03% for English apps and 77.41% for Chinese apps, see Table VIII). Among the hidden privacy settings, 82.16% are set to leak by default, which means that users cannot quickly find them and stop the leakage of their privacy. Such privacy includes all the six categories in Table I. We further counted the number of privacy settings and hidden ones in each category for the 200 apps. The results are shown in Table IX. To our surprise, 100% of users' social connections are exposed by default. Also in this category, about one-third of the settings are hidden, which makes it very hard for users to switch them off.

Further, our participants reported that the texts of some privacy settings are confusing. Especially, some texts look like protecting users' privacy (e.g., the setting "Hide location"), but the setting is switched off by default, which means the location is *not* hidden. Maybe developers want such description to let users feel that the app is protecting users' privacy. Similar texts include *"Do not show people I follow and groups I've joined"* (in the social app *Blued* with over 14 million downloads) and *"Do not show my listening history to friends"* (in the top popular music app *QQ music* with over 550 million downloads in China). We manually analyzed 599 privacy settings in the 200 apps, and found 21.87% of the privacy settings have this problem (20.05% for English apps and 24.07% for Chinese apps). Most users just leave the settings there without changing the default status, unintentionally leaking their privacy.

VI. Suggestions for Developers

Based on our measurement, we find that the problem of hidden privacy settings is severe and pervasive. A good guidance on how to design privacy settings for developers is in urgent need. One may think of the UI design principles from big companies like Google and Apple [18], [28], [29]. However, they only give very rough suggestions such as *"keeping an app consistent by using system-provided interface elements, standard text styles, uniform terminology"*, and more importantly, they do not focus on privacy settings. GSMA [30], an originally-European trade body that represents the interests of mobile network operators worldwide, makes "Privacy Design Guidelines for Mobile Application Development" [3] that requires developers to ensure that defaulting settings are privacy protective and give users control of their personal information in the ways which are easy to understand and use. However, still they fail to provide enough detailed suggestions. To help developers better design their privacy settings, we summarize some suggestions for them based on the human subject studies and measurement.

• **For all privacy settings, let *Settings*→*Privacy Settings* starts their UI-paths.** Most users get used to locating a privacy setting using the two views. In this way, developers can avoid the categorization problems such as placing a privacy setting into an inappropriate view which users may never visit for searching privacy settings. However, for some privacy settings that are also reasonable to show up in other places, we suggest the developers to use multiple entries to connect to the settings from these places. For example, for the setting "share location",

in addition to the entry under *Privacy Settings*, we suggest the developers to add another entry in *Location Settings*.

• **Never put too many UI elements in a view. Instead, use nested views when necessary.** A view containing too many UI elements is one of the root causes of hidden privacy settings (see Section II-B). Therefore, we suggest developers to ensure that all the UI elements in the same view can be displayed in one screen. When too many privacy settings have to be put into a view, developers should consider group them according to their functionalities and create a nested view for the group.

• **Keep the text of a privacy setting short and concise; separate descriptions of the setting from the text.** Some developers like to provide detailed descriptions for a privacy setting, explaining more on the privacy been used. However, users may not read them due to the complexity. Developers should use a short text to describe the basic functionalities of the setting and separate the long description from the text.

• **Use typical icons as indicators and add text descriptions to every icon.** According to the second user study, users are often confused by non-typical icons. What they like are concise text descriptions.

• **Design a privacy setting as follows: switch on for sharing privacy; switch off for stopping sharing.** Users get used to this manner for a privacy setting. Otherwise, some users may leak privacy unintentionally (See Section V-D).

Besides the suggestions, we will also support our tool Hound for developers to identify hidden privacy settings before releasing their apps. Once a hidden setting is identified, developers can refer to our suggestions to improve their UI design.

VII. Limitations

User study. The first limitation of the user study is that the difficulty level of each privacy setting is quantified by the average score given from 5 participants. And the results might be subject to a lack of representativeness due to the relatively small number of participants for each setting. However, the average Fleiss's kappa of our results is as high as 71.93%, indicating a substantial agreement among the responses of the participants. Therefore, the results of our user study are a legitimate evaluation of the hiddenness of privacy settings. Another limitation is that 57.26% of the participants from China are working in computer science or related fields. These participants have better skills for using apps comparing to the general public so that they tend to view a privacy setting more easy-to-find. Thus, having more users with different background will make the result more general. Furthermore, it would be ideal if we can find users that have used all the mobile apps we evaluate. However, it is difficult to recruit such users with the less popular apps. Alternatively, we require participants to use mobile for at least one year to have experienced users. And in our user study, we also ask our participants whether they have used the app before. According to our research, whether using the app or not doesn't have much impact on their ability to set the privacy settings (t-test, $p>0.8$).

Technique. Although the technique *semantics-based UI tracing* is based on the static analysis which cannot handle obfuscation, Hound can still identify the indicator texts and the title of views to extract the UI-paths if an obfuscated app leaves semantics in the resources files such as *.layout* file. We randomly selected 200 apps in our dataset, and only one app (0.5%) is obfuscated entirely without any semantic information in the app. Hound cannot identify privacy settings which are loaded from the Internet dynamically. In our randomly selected 200 apps, only nine (4.5%) apps load their privacy settings from servers each time. This situation is not considered in our research.

VIII. Related Work

Mobile privacy protection. Android system provides a permission-based security model that restricts an app's access to user private data [6], [31], [32], [33]. A lot of previous work [34], [35], [36], [37] focus on how to configure permission preferences to avoid unnecessary permissions application and the privilege escalation issue. Other works [38], [39], [40], [41] provide various approaches to enhance Android permission system for better protection. However, system permissions only protect limited personal data (i.e., the privacy in the first category in Table I such as contacts and location). Our research studies on much broader privacy settings touched by mobile apps beyond the protection by system permissions (i.e., the other five categories in Table I).

Mobile app usability. The usability is essential in app design. Lots of researches [42], [43], [44], [45], [46], [47], [48] work on how to achieve the goal by studying the UI design principles such as considering screen size, text font, data entry methods, etc. But most of them focus on the general UI design for an app, and only very few studies concentrate on the usability of mobile privacy settings [49], [50], [51]. In addition to applying general UI design rules on privacy settings [49], some studies[51], [50] focus on the accuracy of privacy setting text, such as discussing the consistency between the text and the desire of privacy settings [51], and providing the suggestions for designing a better understanding of privacy setting text for users [50]. Different from them, our research is studying the difficulties in locating a privacy setting, which is the first step when considering the usability of privacy settings.

UI-paths analysis. Hidden feature extractor in HPSI is supported by the UI-path tracer, which extracts the UI-paths that link an app's home view to a given privacy setting. Previous studies [52], [53], [54] extract UI-paths by searching Android API *startActivity* to recover the connections among views. But they cannot find the indicator that triggers one view to another. To solve the problem, Hound uses *semantics-based UI tracing* to correlate an indicator with a view according to the semantic information of the indicator and title of the views.

IX. Conclusion

In this paper, we report the first large-scale measurement study on privacy settings that are difficult for users to find. More specifically, we did two user studies to understand users' perceptions of data exposure controlled by privacy settings and found whether these settings are presented to them in the right way. From participants' feedback, we summarized six root causes for the trouble and converted them into 14 features for further detect them. We build a tool called `Hound` with a high accuracy of 93.54% to recover privacy settings and identify those problematic ones, which uses a novel technique named *semantics-based UI tracing* to extract features for training the classifier. Running the Hound on 100,000 apps from Google Play and third-party markets, we find that over one-third (36.29%) of the privacy settings from these apps are hidden and 82.16% of them by default leak out user private data. We observed that the problem of hidden privacy settings becomes more serious from the year 2017 to 2018, possibly due to the fundamental causes of privacy settings' problematic designs. Finally, we provide five suggestions for developers to design privacy settings.

X. Acknowledgment

We would like to thank Roya Ensafi and the anonymous reviewers for their insightful comments. IU authors are supported in part by NSF CNS-1527141, 1618493, 1801432, 1838083, ARO W911NF1610127 and Samsung Gift fund. IIE authors are supported in part by National Key R&D Program of China (No.2016QY04W0805, No.2016QY071405), NSFC U1836211, U1536106, 61728209, National Top-notch Youth Talents Program of China, Youth Innovation Promotion Association CAS, Beijing Nova Program, Beijing Natural Science Foundation (No.JQ18011) and National Frontier Science and Technology Innovation Project (No. YJKYYQ20170070), Strategic Priority Research Program of CAS (No.XDC02000000). UVA authors are supported in part by NSF 1823325 and UVA Research Innovation Award.

References

[1] EDRi, "Big brother awards belgium: Facebook is the privacy villain of the year," https://edri.org/bba-belgium-2016/, 2016.

[2] Y. Nan, M. Yang, Z. Yang, S. Zhou, G. Gu, and X. Wang, "Uipicker: User-input privacy identification in mobile applications." in *USENIX Security Symposium*, 2015, pp. 993–1008.

[3] GSMA, "Privacy design guidelines for mobile application development," 2012.

[4] I. o. i. a. Amazon Mechanical Turk, "amazon mechanical turk," https://www.mturk.com/, 2018.

[5] kim in oregon, "Mturk for academics," https://mturk4academics.wordpress.com/2015/03/13/3-types-of-attention-checks/ , 2018.

[6] G. Android Developer, "System permissions," https://developer.android.com/guide/topics/permissions/index.html, 2018.

[7] appetizeio, "Appetize.io-run native mobile apps in your browser," https://appetize.io/, 2018.

[8] SurveyMonkey, "Open-ended questions: Enrich your data with more context," https://www.surveymonkey.com/mp/open-ended-questions-get-more-context-to-enrich-your-data/ , 2018.

[9] J. L. Fleiss, "Measuring nominal scale agreement among many raters." *Psychological bulletin*, vol. 76, no. 5, p. 378, 1971.

[10] Google, "Layouts," https://developer.android.com/guide/topics/ui/declaring-layout.html, 2018.

[11] "Checkbox," https://developer.android.com/reference/android/widget/CheckBox, 2018.

[12] "Switch," https://developer.android.com/reference/android/widget/Switch, 2018.

[13] "Togglebutton," https://developer.android.com/reference/android/widget/ToggleButton, 2018.

[14] Wikipeida, "Jaccard index," https://en.wikipedia.org/wiki/Jaccard_index, 2018.

[15] G. Android Developer, "Widgets," https://developer.android.com/guide/topics/appwidgets/index.html, 2018.

[16] A. D. Google, "Activity," https://developer.android.com/reference/android/app/Activity.html, 2018.

[17] R. Radhakrishnan, N. Vijaykrishnan, L. K. John, A. Sivasubramaniam, J. Rubio, and J. Sabarinathan, "Java runtime systems: Characterization and architectural implications," *IEEE Transactions on Computers*, vol. 50, no. 2, pp. 131–146, 2001.

[18] Google, "Android design principles," https://developer.android.com/design/get-started/principles.html, 2018.

[19] R. W. Connor Tumbleson, "Apktool," https://ibotpeaches.github.io/Apktool/, 2018.

[20] Python, "Scikit-learn, machine learning in python," http://scikit-learn.org/stable/, 2018.

[21] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[22] E. Brill, "Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging," *Computational linguistics*, vol. 21, no. 4, pp. 543–565, 1995.

[23] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[24] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[25] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[26] J. S. Marc Claesen and D. Popovic, "Optunity," http://optunity.readthedocs.io/en/latest/, 2014.

[27] Google, "Google play," https://play.google.com/store, 2018.

[28] "Material design," https://material.io/guidelines/, 2018.

[29] Apple, "Human interface guidelines," https://developer.apple.com/ios/human-interface-guidelines/overview/themes/, 2018.

[30] GSMA, "Gsma," https://www.gsma.com/, 2018.

[31] Apple, "Requesting permission," https://developer.apple.com/ios/human-interface-guidelines/app-architecture/requesting-permission/, 2018.

[32] A. Bartel, J. Klein, Y. Le Traon, and M. Monperrus, "Automatically securing permission-based software by reducing the attack surface: An application to android," in *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*. ACM, 2012, pp. 274–277.

[33] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android permissions demystified," in *Proceedings of the 18th ACM conference on Computer and communications security*. ACM, 2011, pp. 627–638.

[34] J. Lin, B. Liu, N. Sadeh, and J. I. Hong, "Modeling users mobile app privacy preferences: Restoring usability in a sea of permission settings," 2014.

[35] R. Johnson, Z. Wang, C. Gagnon, and A. Stavrou, "Analysis of android applications' permissions," in *Software Security and Reliability Companion (SERE-C), 2012 IEEE Sixth International Conference on*. IEEE, 2012, pp. 45–46.

[36] Q. Do, B. Martini, and K.-K. R. Choo, "Enhancing user privacy on android mobile devices via permissions removal," in *System Sciences (HICSS), 2014 47th Hawaii International Conference on*. IEEE, 2014, pp. 5070–5079.

[37] M. H. Loorak, P. W. Fong, and S. Carpendale, "Papilio: Visualizing android application permissions," in *Computer Graphics Forum*, vol. 33, no. 3. Wiley Online Library, 2014, pp. 391–400.

[38] Y. Wang, J. Zheng, C. Sun, and S. Mukkamala, "Quantitative security risk assessment of android permissions and applications," in *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, 2013, pp. 226–241.

[39] D. Barrera, H. G. Kayacik, P. C. van Oorschot, and A. Somayaji, "A methodology for empirical analysis of permission-based security models and its application to android," in *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 2010, pp. 73–84.

[40] A. Armando, R. Carbone, G. Costa, and A. Merlo, "Android permissions unleashed," in *Computer Security Foundations Symposium (CSF), 2015 IEEE 28th*. IEEE, 2015, pp. 320–333.

[41] M. Nauman, S. Khan, and X. Zhang, "Apex: extending android permission model and enforcement with user-defined runtime constraints," in *Proceedings of the 5th ACM symposium on information, computer and communications security*. ACM, 2010, pp. 328–332.

[42] C. Coursaris and D. Kim, "A qualitative review of empirical mobile usability studies," *AMCIS 2006 Proceedings*, p. 352, 2006.

[43] R. Harrison, D. Flood, and D. Duce, "Usability of mobile applications: literature review and rationale for a new usability model," *Journal of Interaction Science*, vol. 1, no. 1, p. 1, 2013.

[44] D. Balfanz, G. Durfee, D. K. Smetters, and R. E. Grinter, "In search of usable security: Five lessons from the field," *IEEE Security & Privacy*, vol. 2, no. 5, pp. 19–24, 2004.

[45] N. Asokan and C. Kuo, "Usable mobile security," in *International Conference on Distributed Computing and Internet Technology*. Springer, 2012, pp. 1–6.

[46] C. Ryan and A. Gonsalves, "The effect of context and application type on mobile usability: an empirical study," in *Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38*. Australian Computer Society, Inc., 2005, pp. 115–124.

[47] G. Buchanan, S. Farrant, M. Jones, H. Thimbleby, G. Marsden, and M. Pazzani, "Improving mobile internet usability," in *Proceedings of the 10th international conference on World Wide Web*. ACM, 2001, pp. 673–680.

[48] J. Gong and P. Tarasewich, "Guidelines for handheld mobile device interface design," in *Proceedings of DSI 2004 Annual Meeting*, 2004, pp. 3751–3756.

[49] T. Paul, D. Puscher, and T. Strufe, "Improving the usability of privacy settings in facebook," *arXiv preprint arXiv:1109.6046*, 2011.

[50] N. Wang, P. Wisniewski, H. Xu, and J. Grossklags, "Designing the default privacy settings for facebook applications," in *Proceedings of the companion publication of the 17th ACM conference on Computer supported cooperative work & social computing*. ACM, 2014, pp. 249–252.

[51] Y. Liu, K. P. Gummadi, B. Krishnamurthy, and A. Mislove, "Analyzing facebook privacy settings: user expectations vs. reality," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*. ACM, 2011, pp. 61–70.

[52] C. Zheng, S. Zhu, S. Dai, G. Gu, X. Gong, X. Han, and W. Zou, "Smartdroid: an automatic system for revealing ui-based trigger conditions in android applications," in *Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices*. ACM, 2012, pp. 93–104.

[53] F. Zhang, H. Huang, S. Zhu, D. Wu, and P. Liu, "Viewdroid: Towards obfuscation-resilient mobile application repackaging detection," in *Proceedings of the 2014 ACM conference on Security and privacy in wireless & mobile networks*. ACM, 2014, pp. 25–36.

[54] K. Chen, P. Wang, Y. Lee, X. Wang, N. Zhang, H. Huang, W. Zou, and P. Liu, "Finding unknown malice in 10 seconds: Mass vetting for new threats at the google-play scale." in *USENIX Security Symposium*, vol. 15, 2015.

APPENDIX

A. Supplying documents for Surveys

To see the example survey, please visit: https://sites.google.com/view/mobi-privacy/home/user-study.

B. Tables and Figures

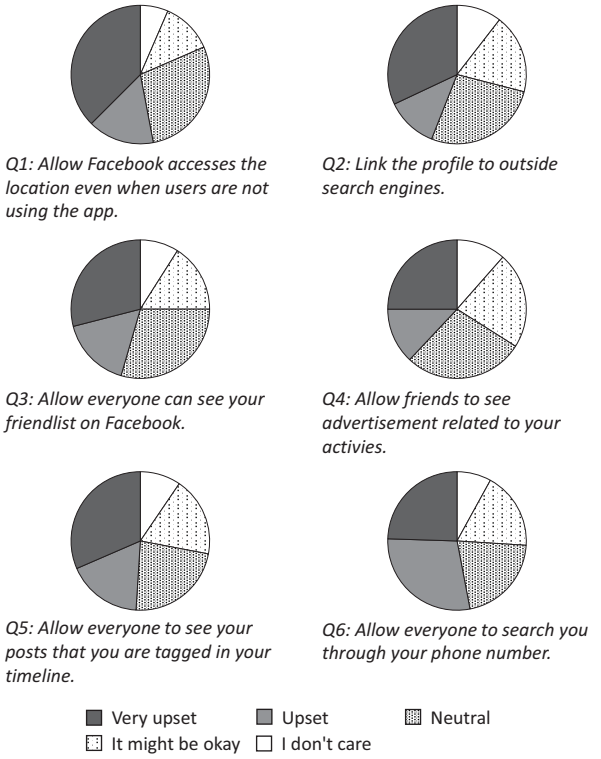

Fig. 12: Distributions of the user perspectives to six privacy settings in the first user study

Fig. 13: Example of hidden privacy setting in Alipay caused by a long UI-path

TABLE X: Calculation of features

p is a given UI-path

Feature
$F_1(p) = \underset{t \in T}{max}\{Jaccard(N_t, N_p)\}$, where T is the typical UI-path set, N_i is the node set of a UI-path i.
$F_2(p) = \begin{cases} 1 & (\forall x, x \in T_p, x \neq \textit{“Privacy Settings”}) \; and \; (\exists v, v \in A, H_v = \textit{“Privacy Settings”}) \\ 0 & \text{Otherwise} \end{cases}$, where T_i is the indicator text set of a UI-path i, A is the view set of the app, H_i is the title of a view i.
$F_3(p) = \begin{cases} 1 & (\forall x, x \in T_p, x \neq \textit{“Settings”}) \; and \; (\exists v, v \in A, H_v = \textit{“Settings”}) \\ 0 & \text{Otherwise} \end{cases}$, where T_i is the indicator text set of a UI-path i, A is the view set of the app, H_i is the title of a view i.
$F_4(p) = \underset{t \in T_p}{avg}\{\underset{m \in M}{max}\{Jaccard(W_t, W_m)\}\}$, where T_i is the indicator text set of a UI-path i, M is the most common indicator text set, W_i is the word set of a text i.
$F_5(p) = \sum_{t \in T_p} f(t)$, where T_i is the indicator text set of a UI-path i, $f(i) = \begin{cases} 1 & \text{if } i \text{ is related to privacy} \\ 0 & \text{otherwise.} \end{cases}$
$F_6(p) = \sum_{c \in C_p} f(c)$, where C_i is the icon set of a UI-path i, $f(i) = \begin{cases} 1 & \text{if } i \text{ is a gear icon or a portrait icon} \\ 0 & \text{otherwise.} \end{cases}$
$F_7(p) = \lvert C_p \rvert$, where C_i is the icon set of a UI-path i.
$F_8(p) = \lvert V_p \rvert$, where V_i is the view set of a UI-path i.
$F_9(p) = \lvert T_{K_p} \rvert$, where K_i is the key view of a UI-path i, T_i is the indicator set of a view i.
$F_{10}(p) = \dfrac{\sum_{t \in T_{K_p}} f(t)}{F_9(p)}$, where K_i is the key view of a UI-path i, T_i is the indicator text set of a view i, $f(i) = \begin{cases} 1 & \text{if } i \text{ is related to privacy} \\ 0 & \text{otherwise.} \end{cases}$
$F_{11}(p) = \underset{v \in V_p}{max}\{\lvert T_v \rvert\}$, where V_i is the view set of a UI-path i, T_i is the indicator set of a view i.
$F_{12}(p) = f(S_p, K_p)$, where S_i is the privacy setting of a UI-path i, K_i is the key view of a UI-path i, $f(i, j)$ is the position of the setting i from the top of the view j.
$F_{13}(p) = \underset{v \in V_p}{avg}\{f(S_{v,p}, v)\}$, where V_i is the view set of a UI-path, $S_{i,j}$ is the indicator of a UI-path j in the view i, $f(i, j)$ is the position of the setting i from the top of the view j.
$F_{14}(p) = \lvert W_{T_p} \rvert$, where T_i is the privacy setting text of a UI-path i, W_i is the word set of a text i.