Crowdsourced Exploration of Security Configurations

Qatrunnada Ismail[†] **Tousif Ahmed**[†] **Apu Kapadia**[†] **Michael K. Reiter**[‡]
[†]School of Informatics and Computing
Indiana University Bloomington
Bloomington, IN, USA
[‡]Department of Computer Science
University of North Carolina
Chapel Hill, NC, USA
{qismail, touahmed, kapadia}@indiana.edu, reiter@cs.unc.edu

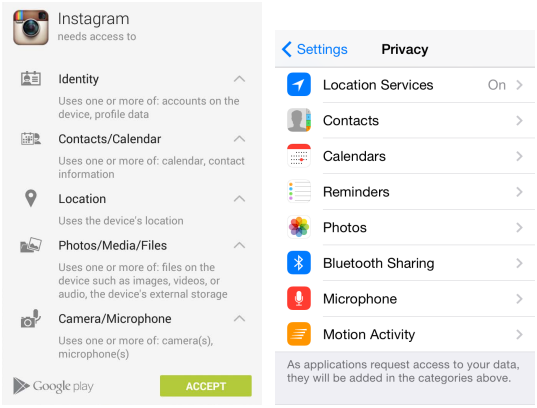
ABSTRACT
Smartphone apps today request permission to access a multitude of sensitive resources, which users must accept completely during installation (e.g., on Android) or selectively configure after installation (e.g., on iOS, but also planned for Android). Everyday users, however, do not have the ability to make informed decisions about which permissions are essential for their usage. For enhanced privacy, we seek to leverage crowdsourcing to find minimal sets of permissions that will preserve the usability of the app for diverse users.

We advocate an efficient 'lattice-based' crowd-management strategy to explore the space of permissions sets. We conducted a user study ($N = 26$) in which participants explored different permission sets for the popular Instagram app. This study validates our efficient crowd management strategy and shows that usability scores for diverse users can be predicted accurately, enabling suitable recommendations.

Author Keywords
Android apps; crowdsourcing; permissions; privacy

ACM Classification Keywords
H.5.m Information interfaces and presentation (e.g., HCI): Miscellaneous; D.4.6 Software: Security and Protection—*Access Control*

INTRODUCTION
Users of mobile devices such as smartphones and tablets are able to install apps from online marketplaces that feature millions of apps.[1] These apps can potentially access various sensors (e.g., accelerometer, microphone, GPS, and camera) and user data (e.g., contact information and photos). In general, it is difficult for the operating system (such as Android or iOS) to ascertain whether an app is making legitimate or nefarious uses of the sensors or data. Instead, during the installation of an app, the Android operating system displays all the requested permissions, which the user either approves or denies as shown in Figure 1a. On iOS, users initially install apps and then are able to selectively enable or disable privacy sensitive permissions as shown in Figure 1b. Google has been experimenting with a similar feature ("App Ops") that allows users to selectively disable specific permissions, but it has not yet been released officially.[2]



(a) Android **(b) iOS**

Figure 1. Control panels related to application permissions.

Ideally users would determine the fewest permissions needed for an app to support their regular usage of the app (i.e., the 'principle of least privilege' [14]). For example, Angry Birds requests access to location, but may work well for most users if access to location is disabled. While several research proposals have been made to allow more granular control or detect anomalous uses of such sensors (Bugiel et al. [2] provide a comprehensive overview of related work on Android OS security), it remains unclear how users can decide which permissions are useful for the functioning of the app. Felt et al. [4] proposed an automated technique to determine which permissions are *not* used by an app, but they did not address the relative importance of each permission that *is* used by the app. While users can provide feedback through ratings and comments on App marketplaces like Apple's App Store and the Google Play Store, no feedback or ratings are available on various *configurations* (i.e., with various combinations of

[1] `http://www.appbrain.com/stats/number-of-android-apps/`, `http://toucharcade.com/2014/06/02/new-ios-8-app-store-features/`

[2] `https://www.eff.org/deeplinks/2013/12/google-removes-vital-privacy-features-android-shortly-after-adding-them/`

CHI 2015, Apr 18–23, 2015, Seoul, Republic of Korea
ACM 978-1-4503-3145-6/15/04. http://dx.doi.org/10.1145/2702123.2702370

permissions disabled) of the app. Indeed, there are too many possible configurations of an app, and it is not clear if it feasible to provide meaningful feedback about all configurations.

We advocate a crowdsourced approach that leverages humans to efficiently assess various security configurations for an app and identify app configurations that strike tradeoffs between privacy and usability suitable to each user. After gaining familiarity with an app, each crowd member would volunteer to utilize a small number of configurations for that app over a test period. Based on the usability scores of configurations given by a specific user, similar crowd members would be identified and then used to recommend configurations suitable to that user. We envision that our system can be easily integrated into popular app marketplaces such as the Apple App Store and the Google Play Store. Instead of rating only one configuration of an app, smartphone users would test and rate different configurations of apps. Then our system would recommend appropriate configurations to users based on their similarities to other users in the marketplace, i.e., the crowd.

The proposed approach, however, poses challenges. First, as the number n of permissions requested by an application increases, the number of possible permission sets increases exponentially (2^n). Thus a scalable approach is needed to collaboratively explore the permission space. Second, even if these configurations can be collaboratively explored, it is not clear whether suitable recommendations can be made to a diverse set of users. A permission set that is acceptable to some users may not be acceptable to others. Finally, unobtrusively deploying and assessing configurations would benefit from measures of 'usability' that do not involve user intervention.

Research questions. This research seeks to leverage crowdsourcing to efficiently find a set of permissions for each user installing an app that strikes an acceptable balance between app usability and user privacy (by finding the permissions that are infrequently used by users or permissions that users do not mind losing). More specifically, we seek to answer the following research questions:

R1: *Can we use crowdsourcing scalably to explore security configurations of an app?* Using crowdsourcing, we seek to obtain users' perspectives on the usability of the app when certain combinations of permissions are disabled. Based on the intuition that usability scores cannot generally increase when additional permissions are removed, we develop a 'lattice-based' approach to prune the search space of permission sets and evaluate its efficacy.

R2: *Can we recommend suitable permission sets based on the crowd's ratings?* We would like to predict the suitability of various app configurations for users based on ratings obtained from the crowd (other users). In particular we explore whether collaborative filtering can be used to make such predictions effectively.

R3: *What proxies can be used to determine the usability of an app without asking users for ratings explicitly?* We seek to find if we can ascertain the 'usability' of an app in automated ways, such as measuring the time spent by a user using an app or logging any problems s/he encounters. Automated measures require fewer user interruptions and so could permit a less obtrusive exploration of the state space.

Our contributions. Before the suggested crowdsourcing strategy can be applied in general, we seek to validate our lattice-based approach and the viability of making accurate predictions of usability scores. As a first step, we perform a user study ($N = 26$) on one specific, but popular, app (Instagram). Participants tried different configurations of this app in which each configuration had specific permissions removed. Participants reported their feedback about each configuration and answered exit surveys at the end of the study.

We found that: 1) as more permissions were removed, the reported usability scores dropped (or did not increase) significantly, validating the lattice-based approach; 2) based on usability scores given by the crowd, we can predict participants' individual ratings for various subsets of permissions, and thus make suitable recommendations; and 3) to measure usability, instead of asking users for explicit feedback, other attributes such as the usage time of an app and the number of problems encountered are suitable proxies for usability.

RELATED WORK

Crowdsourcing

Crowdsourcing has been used in many studies related to Android privacy and security. Amini et al. [1] developed AppScanner, which analyzes and learns application behaviors to find privacy-related issues. In their crowdsourcing step, they ask users for their reactions to behaviors uncovered by their tool. Crowdsourcing was also used by Lin et al. [10] to capture users' expectations about privacy-related behaviors of Android applications. They used their findings to design a technique to inform Android users about the way permissions are used by an application. However, it may not be clear to users how removing certain permissions may affect the usability of the app. In our crowdsourcing step, instead of only asking users about their *expectations*, we provide them with different configurations of applications (with different permissions removed) and get users' feedback based on their *experience* with those configurations.

Studying the Effects of Permission Removal

Since users do not have fine-grained control over what specific permissions they grant apps in Android, researchers have proposed systems to give users the ability to do so. For example, Nauman et al. [12] modified Android with an advanced application installer that enables users to allow or deny each individual permission requested at installation. Rather than asking users to evaluate permissions at installation, we instead ask them to evaluate an application with selected permissions disabled. We then use their feedback as a basis for performing collaborative filtering to benefit other users. Moreover, our method of disabling permissions does not modify Android.

There are also studies about the effects on applications' performance after removing permissions. Notably, Kennedy et al. [9] proposed a system that removes permissions from Android applications to determine which removed permissions

cause the app to crash. Their system, however, focuses on an automated process for evaluating the run-time effects of removing permissions and does not assess the impact on users based on real-world usage, as we do.

App Ops, a feature in Android v4.3, allowed users to selectively disable permissions for apps on their phones. However, Google removed this feature in the next update, reporting that it was experimental and could cause apps to behave in unexpected ways. For our approach, we assume the apps being explored have reasonable exception handling. We expect that App Ops-like functionality will eventually be released for Android (as it is already available in iOS) and that most apps will handle permissions restrictions gracefully.

Users' Perspectives about Permissions

Felt et al. [5, 6] and Kelly et al. [7] showed that the current way of displaying permissions is not clear to users and not effective at informing them about the potential risks of installing an app. Kelly et al. [8] thus proposed a more effective way to display the permissions and found it to be helpful for users to make better decisions with respect to their privacy. Rather than asking users directly if they think a specific permission is needed, we let users test app configurations with some permissions removed and report their feedback. We expect a combination of these approaches would be useful, whereby useful suggestions through crowdsourcing can improve information provided in the installation interface.

Collaborative Filtering

Recommender systems are used to predict ratings and to create personal recommendations in various domains. By applying statistical methods and knowledge discovery techniques, it is possible to recommend products based on user preferences [15]. Collaborative filtering is a recommendation technique that is used for predicting a user's ratings for unknown products based on the user's previous ratings and aggregating other (similar) users' ratings. Two common techniques for collaborative filtering are 'user-based' collaborative filtering and 'item-based' collaborative filtering. In user-based collaborative filtering, a user's rating for any particular item is predicted from ratings by similar users for that item [13]. In item-based collaborative filtering, ratings for an item are predicted from ratings for similar items [16]. In this paper, we incorporate the idea of user-based collaborative filtering into the mobile app privacy domain and use it for predicting the suitability of various permission sets.

APPROACH

In this section, we describe our proposed crowdsourcing approach, usability metrics, and recommender system. Moreover, we present the hypotheses that we test in our user study.

Lattice Based Approach for Crowd Management

We propose an approach whereby crowd members strategically explore the various permission sets of an app in a way that is scalable with the number of permissions requested by the app. Our main hypothesis here is that as the set of permissions removed grows, the 'usability' of the app does not increase. (We defer the discussion about how exactly we measure 'usability' to the next subsection.) Intuitively, if a set of permissions is made more restrictive, we expect the number of features of the app to also be reduced and thus result in a lower or equal rating.[3] For example, if a user thinks that a specific app is unusable when location permission is removed, s/he is likely to think the same (or worse) if the location and camera permissions are removed together. This intuition suggests an approach of a structured exploration of the permission space as we formalize next.

The permission space can be represented by a lattice structure. Consider a lattice structure as shown in Figure 2 for an app that requests four permissions. The nodes in the lattice represent the sets of *removed* permissions. The null vertex represents the original app where there are no permissions removed, so it is not shown in the figure. Each level in the lattice going upwards represents an increasing number of permissions removed. So, the first level represents nodes with one permission removed, the second level has nodes with two permissions removed, and so on.

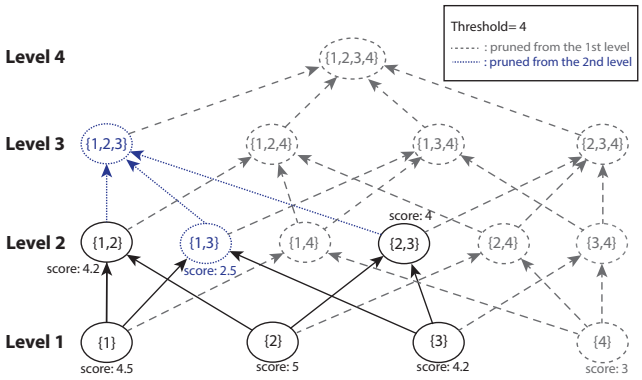


Figure 2. The lattice structure where each node represents specific permissions that are removed.

We suggest the following crowd-based exploration strategy: First, the crowd explores the first level in the lattice and the average usability score is calculated for each node. If the scores of any of these nodes fall below a certain threshold, their ancestors can be pruned and not explored because the scores are expected to be non-increasing. For example, if we consider the lattice in Figure 2, after testing all four nodes in the first level, the node $\{4\}$ has a score below the threshold. So, we prune all of this node's ancestors. This leaves us with fewer nodes to test as we proceed upwards. After testing the nodes $\{1,2\}$,$\{1,3\}$ and $\{2,3\}$, we may find that the average score for $\{1,3\}$ falls below the threshold, so we prune its ancestor $\{1,2,3\}$. This approach enables us to explore the

[3] Of course, this is not true as a rule. For example, removing one permission alone might cause the app to consistently crash, though in addition removing a second permission might prevent the app from ever attempting to use the first (and so from crashing). Though an artificial example, it shows that removing permissions need not monotonically decay the usability of the app. That said, recall that we assume apps will handle the removal of permissions gracefully; presumably this should already be true for iOS apps and will increasingly become so for Android apps once App Ops is (re)released.

permission space efficiently and find suitable configurations of permissions based on the crowd's reports, i.e., nodes {1,2} and {2,3} in our example. This technique can be more efficient than exploring the whole space, especially when the number n of permissions is large.

Before this strategy can be applied in practice, it is important to ascertain whether the presumed lattice-based relationship of usability scores holds. As a first step, we evaluate the following hypothesis for a specific application (Instagram):

H1: The usability scores of the nodes in the lattice are non-increasing as we proceed upwards in the lattice and remove more permissions.

Measuring Usability

In this paper we use the term "usability" to refer to how 'acceptable' users consider an application to be when some of its advertised functionality has been intentionally restricted. We obtain this 'ground truth' information by asking participants about their level of acceptance of using a configuration of the app (until the next major app update) with specific features disabled. We also ask them to rate the different configurations compared to the original app in which no permissions are removed. In our envisioned crowdsourcing system, we seek to measure the usability of an app indirectly without user intervention by using indirect measures that can serve as a proxy for usability. Those measures include the time spent using an app and whether there are problems found while using it.

Therefore, we have the hypotheses mentioned below. We also test whether the 'ratings' provided by users agree with the 'acceptability' scores (lending validity to using 'acceptability' as a ground-truth measure):

H2: Under the assumption that the original app is perceived as usable by users, a user's rating of a customized configuration of the app as better or worse than the original is a good predictor for the whether the user finds the customized configuration acceptable.

H3: The time spent using a customized configuration of the app can be used as a proxy to measure its usability. This means that participants who use a particular configuration longer than another configuration find the former configuration more acceptable.

H4: Whether users find problems in a customized configuration of an app can be used as a proxy to measure its usability. Participants who identify more problems in a configuration of an app will rate it as less usable than a configuration for an app where fewer problems are encountered.

Recommender System

Our goal is to leverage the crowd-based exploration of the permission subsets to recommend a suitable one for a particular user. We advocate a 'user-based collaborative filtering' approach [13] to recommend permission subsets to users based on their similarity (of ratings) to other users in the crowd.

The user-based collaborative filtering computes the predicted ratings for user $u \in U$ on all 'items' $\in I$ where the items represent permission subsets that u did not try. The algorithm first identifies the most similar users to u based on similarity in their ratings for the common items that they have tried. Those users are denoted by *k-nearest neighbors* where k is the number of those similar users. The predicted rating for user u on any item $j \in I$ is then computed as [3]:

$$p_{u,j} = \bar{r}_u + \frac{\sum_{v \in \text{Neighbors}(u)} s(u,v)(r_{v,j} - \bar{r}_v)}{\sum_{v \in \text{Neighbors}(u)} |s(u,v)|}$$

where $\bar{r}_u$ is the mean rating for user u (across all items rated by that user), Neighbors(u) is the list of u's similar users, $r_{v,j}$ is the usability rating of user v for permission set j (a positive integer in our case), and $s(u,v)$ is the 'similarity' between user u and his neighbor v. This formula calculates the difference between each neighbor v's rating for item j and v's mean rating for all items, takes the weighted mean of this difference across neighbors (using a similarity measure as the weight), and then applies this difference to u's mean rating. We give an example in the Findings section (see Table 3).

The similarity measure $s(u,v)$ can be computed, e.g., using the 'cosine similarity' of the users' rating vectors, which is shown in the formula bellow. $\mathbf{r}_u$ and $\mathbf{r}_v$ are users' u and v ratings vectors and the unknown ratings are replaced by 0.

$$s(u,v) = \frac{\mathbf{r}_u \cdot \mathbf{r}_v}{\|\mathbf{r}_u\| \, \|\mathbf{r}_v\|} = \frac{\sum_{i=1}^{n} r_{u,i} r_{v,i}}{\sqrt{\sum_{i=1}^{n} r_{u,i}^2} \sqrt{\sum_{i=1}^{n} r_{v,i}^2}}$$

We propose a specialized approach based on the lattice structure. For example, we may want to recommend as restrictive a permission set as possible (higher up in the lattice, which means enhanced privacy) as long as the usability score does not fall below a certain threshold. Furthermore, the lattice-based structure can be leveraged to improve the accuracy of our predictions given the relationship between nodes in the lattice. As hypothesized in H1, as we remove more permissions, the usability scores of the apps are non-increasing. So, if the predicted score of any node in the lattice is greater than any of its children scores, we set the predicted score for that node to the minimum score of the children nodes. For example, as we can see from the lattice depicted in Figure 2, node $\{1,2\}$ has two children $\{1\}$ and $\{2\}$. If the predicted score for node $\{1,2\}$ is 5.5, our algorithm sets it to 4.5 which is the minimum score of node $\{1\} = 4.5$ and $\{2\} = 5$. We refer to this approach as the "clamping technique", since the recommended score is clamped to the highest value allowable by the user's own scores in the lattice.

We evaluate how accurately the suggested collaborative filtering approach can be used to predict usability scores for users, and thus the viability of a crowdsourcing-based approach for identifying configurations of applications that balance usability and privacy for individual users.

STUDY METHODOLOGY

System Implementation

To answer our research questions, we conducted a user study in which we focused on Instagram, which is a popular social networking and photo/video sharing Android app. Throughout the paper, we refer to Instagram as the "test app". In the

study, participants installed an Android app called "the ratings app", to provide them with different configurations of the test app and collect their feedback on each configuration.

Choice of the test app. The test app, Instagram, is a photo/video sharing and social networking app. We chose this specific app because it is one of the popular free Android apps in the Google Play store. Besides being popular, the test app requests several permissions categorized as "dangerous" (according to Android's classification),[4] providing access to sensitive user data such as their location and the camera. As permissions in Android apps are divided into four different levels depending on their risk level, we focus on the permissions in the 'dangerous' level. Permissions on this category are displayed to users upon installation of an app and need to be approved to complete the installation process [5]. In our study, we focused on four such permissions providing access to: (1) location, (2) contacts, (3) camera, and (4) microphone.

Customization of the test app for exploration. We used the Android-apktool[5] to unpack the test app's Android application package file (apk file) and access the app's manifest file. The manifest file includes the list of requested permissions. We then created different customized configurations of the test app by removing one or more permissions from the manifest file and then recompiling and repackaging the apk files. We did not have access to the the test app's code itself, and we made no modifications to the application code.

We created all possible configurations of the app by removing one or more of the four permissions listed above. This gave us a total of 15 customized configurations of the test app. The configurations are shown in Figure 4.

The ratings app. When participants installed the ratings app on their phones, they were prompted to install one of the customized configurations of the test app after they removed the original app. After installing a customized configuration of the test app, participants had two days to use it; a timer displayed the time left to request the next configuration of the test app as shown in Figure 3. Also, participants were allowed to ask for a different configuration of the test app at any time before the timer ended, but with a $0.50 deduction from their overall compensation. Moreover, the amount of money earned in the study was displayed by the ratings app as shown in Figure 3. As such, our compensation scheme (described in more detail in the Study Procedure Section) motivated participants to think carefully before requesting a new configuration.

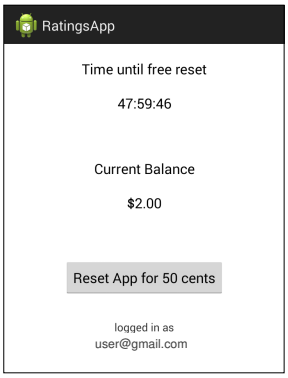


Figure 3. The ratings app home screen

[4] **http://developer.android.com/guide/topics/manifest/permission-element.html**

[5] **https://code.google.com/p/android-apktool/**

Collecting participants' feedback. At the end of the two-day period for exploring a configuration of the test app, participants received notifications to both their phones and email to remind them to visit the ratings app and request the next configuration of the test app. When participants requested a new configuration by clicking on the reset button, a survey dialog sought their feedback about the configuration that they just used. The survey questions are shown in the Appendix, which we summarize here:

Q1. We asked participants about whether they encountered problems in the configuration they just used.
Q2. We listed the disabled features (based on the removed permissions) and asked participants to rate, on a 7-point Likert scale, how acceptable they found this configuration (if they had to use it until the next major app update).
Q3. We asked participants to rate the test app's configuration compared to the original one in which there were no permissions removed.
Q4. We asked participants to estimate the amount of time they used that configuration of the test app.

After answering the questionnaire, participants were given a different configuration of the test app and the timer was reset.

Back-end. When participants asked for a different configuration of the test app, the ratings app connected to a remote server in which participants' answers to the questionnaire described above were recorded and a customized configuration of the test app was retrieved. One of the customized configurations was chosen at random while making sure that it was not previously used by the requesting participant. We also set a threshold for the number of times that each configuration was tested to make sure that we did not end up with some configurations used much more than others. Moreover, to make sure that participants actually used the provided configuration of the test app, the Ratings App kept track of whenever participants used the test app on their phones. This information was recorded at the back-end.

Design Rationale

Minimizing learning effects. The collaborative filtering approach generates recommendations by analyzing multiple ratings from each participant. Thus we needed each participant to test multiple app configurations, as they would in a real system. To minimize learning effects due to repeated exploration by the same participant, we randomized the app configurations, and their order, tested by each participant. Thus participants could not predict whether subsequent configurations would be more (or less) restrictive, and which permissions would be removed.

Managing participant workload. In collaborative filtering, more ratings from a single user generally produce better recommendations for that user (e.g., getting better movie recommendations from Netflix as the viewer watches and rates more movies). We did not want to induce fatigue in our participants (e.g., assigning too many configurations to each participant) and wanted to ensure that participants had enough time to explore each configuration to make an informed decision. Based on a pilot study with two undergraduate stu-

dents, we determined that 5 configurations over 10 days was a reasonable workload to meet both goals, and we devised a compensation scheme as discussed in the next section based on this workload. We note that in our exit survey, 20 out of 26 participants said they felt they had enough time to explore each of their assigned configurations.

Study Procedure

Enrollment. Participants were required to be at least 18 years old, to have lived in the United States for at least five years, to have an Android phone with a data plan, and to already be using Instagram. Qualified participants reported to our lab to enroll in the study. They were informed of the purpose and process of the study. After consenting, participants installed the ratings app. We provided two demo configurations of the test app, allowing participants to familiarize themselves with the ratings app and the study process. We then instructed the participants to contact us after completing the study.

Ethical considerations. Our study was IRB-approved, and Indiana University's General Counsel approved our modification of Instagram's manifest file based on a 'Fair Use' analysis of its license agreement.

End of study questionnaire. At the end of the study, participants were given a questionnaire that asked for their technical backgrounds, demographics, and feedback. It also included questions related to assessing privacy attitudes.

Compensation and incentive scheme. Study participants were paid up to $25 upon the completion of the study. The study was designed to last approximately 10 days and participants earned $2 per day. However, this amount could be affected in the following situations:

- If the participant was supposed to reset the test app at the end of the two-day period but did not do so for a whole day, s/he did not earn the $2 for that day. We used this as an incentive to motivate participants to continue trying different configurations of the test app throughout the study.
- If the participant reset the test app before the end of the two-day period in which s/he was to use the test app, $0.50 was deducted from his/her overall balance. We used this penalty to make participants think twice about whether the current configuration was tolerable. So, if a participant was trying to use a feature that had been disabled, whether s/he asked for a new configuration and lost $0.50 or waited depended on how important this feature was to him/her.

At the end of the study, if no money was deducted from the participant's balance, s/he earned $20. An extra $5 was added if s/he answered the end-of-study questionnaire, which resulted in a maximum compensation of $25.

FINDINGS

Participants

We recruited our participants from Bloomington, IN, USA (a college town that is home to Indiana University) by using flyers, online university classifieds advertisements, and student mailing lists. The study lasted for two months (Feb–Mar 2014). Overall, 30 participants enrolled in our study, 4 of which did not complete the study (testing fewer than 5 configurations) so we removed their ratings. Therefore, in our analysis, we considered the 26 participants who finished the study and we ended up with a total of 137 ratings. In our sample, most of the participants (22 participants) tested 5 configurations of the test app, 3 participants tested 6 configurations and 1 participant tested 9. For the participant who tested 9, he/she repeatedly received a configuration with the camera permission removed and thus requested more configurations than most. Each configuration of the test app was rated by at least 8 participants and at most 11 participants.

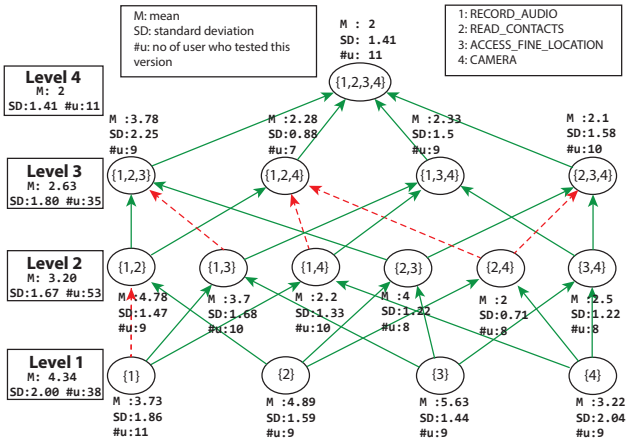


Figure 4. The lattice structure where each node represents specific removed permissions. The red arrows indicate slight increases in scores which are not statistically significant.

Among these 26 participants, 15 were females (57.7%) and 11 were males (42.3%). All but one were students. Seven (26.9%) were Informatics or Computer Science majors, while the rest came from diverse, non-technical majors.

Analysis of the Lattice Structure

Figure 4 shows the average acceptance score (M), standard deviation (SD), and number of participants (#u) per test app configuration as well as aggregated per level in the lattice. Our main hypothesis about the acceptance scores (as a usability measure)—i.e., that they are generally not increasing as we remove more permissions—is evidenced in Figure 4.

Figure 5 also shows that as we remove more permissions, the acceptance scores tend to decrease or stay the same. The big circles in Figure 5 indicate whether the participant encountered a problem or not (Q1 in the survey).

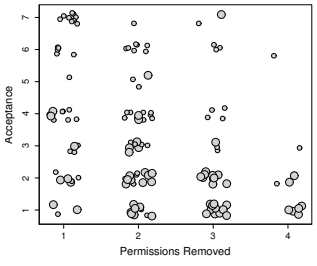


Figure 5. Scatter plot showing the effects on acceptance scores as we remove more permissions.

To test our first hypothesis (H1), we considered using a multi-level model such as the linear mixed effects model which takes into

account the effects of having users contributing multiple data points. However, the homoscedasticity assumption is violated in our data because the variances of the acceptance scores of different permission subsets are not equal; therefore, we used non-parametric tests. Our conclusions hold using both the linear mixed effects model and the non-parametric tests but we report the latter because they are typically more conservative, making fewer assumptions.

We first ran a Kruskal-Wallis test, which is a non-parametric one-way (single factor) ANOVA. This test shows if there is a statistically significant difference between variables but it does not provide information about the nature of the difference (increase or decrease). To examine the difference, we ran a follow-up post hoc test. We ran the non-parametric version of the t-test called the Wilcoxon rank-sum test and since this is a post-hoc test that involves multiple comparisons, we used the Benjamini-Hochberg method to adjust the p-values from the test to control the false discovery rate. The result of Kruskal-Wallis test ($\chi^2 = 21.48$, df $= 3$, $p < 0.001$) indicates that there is significant difference in participants' acceptance scores due to the number of permissions removed. The results of the Wilcoxon rank-sum post-hoc test are shown in Table 1 where the parameter W and the p-value assess the significance of the difference (the decrease) between the variables (# of permissions removed).

Table 1 shows that as more permissions are removed, the acceptance scores decline significantly until 4 permissions are removed (at which point there is no significant difference in the scores compared to removing 3 permissions). This means that as we go from level 3 to 4, the acceptance values are neither increasing nor decreasing significantly. One reason could be that we have few data points for level 4, where we have only one node compared to the other levels. Given that the significant difference between levels indicates a significant decrease, we can reject the null hypothesis and accept our first hypothesis (H1) that the acceptance scores are non-increasing as we remove more permissions.

Comparison of # permissions removed	W	p-value	Adjusted p-value
1 vs. 2	1336.0	0.009**	0.015*
1 vs. 3	990.0	< 0.001***	0.002**
1 vs. 4	349.0	0.001**	0.002**
2 vs. 3	1143.0	0.024*	0.029*
2 vs. 4	423.5	0.017*	0.025*
3 vs. 4	233.5	0.275	0.275

Statistical significance: *** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

Table 1. Wilcoxon rank-sum results for the statistical significance of the decrease in acceptance scores as we remove more permissions.

In addition to the Kruskal-Wallis test, we performed a correlation analysis using Spearman's Rank Coefficient, which measures the tendency for two variables to increase or decrease together. While the p-value tells us the significance of the correlation between the variables, ρ tells us whether the variables are monotonically increasing or decreasing and the degree of monotonicity. Testing the acceptance score versus the number of permissions removed, we found a significant negative relationship ($\rho = -0.39$, $p < 0.001$), indicating that as more permissions are removed, the participants' acceptance scores decline; this supports the results of the previous test.

Special cases. Though we have shown that the acceptance scores are non-increasing as we move upward in the lattice, there are some exceptions in our data, shown by dashed red edges in Figure 4. However, based on Wilcoxon rank-sum tests, these increases are not significant ($p > 0.05$).

Lattice pruning approach. As this is an exploratory study, we focused on four permissions to be able to explore all subsets of permissions (all nodes in the lattice). We thus could test whether we can use the proposed pruning strategy on the lattice instead of testing all nodes in the lattice individually. Based on our questionnaire, an acceptability score of 4 can be chosen as the usability threshold, which means any node with acceptance value below 4 is not acceptable and thus its ancestors can be pruned. While we have tested only one specific app, these results point to the viability of a lattice-based approach for crowd management. This particular lattice suggests that configurations that disable access to {contacts}, {location}, and {contacts, location} are promising configurations that trade off usability and privacy for Instagram. Moreover, after the first level, only the {contacts, location} configuration would have been explored by the strategy.

Rating and Acceptance Scores

In the Methodology section, we summarized the survey questions that we asked participants after using each configuration of the test app. While in Q2 we asked about the participant's acceptance of using the app knowing that specific features were disabled, in Q3 we asked participants to rate the used configuration compared to the original app. Both questions measure, in different ways, how much participants like the app they used. Our data indicates that there is agreement between the two variables which means that the higher the rating, the higher the acceptance score. We used Spearman's Rank Correlation test to see if acceptance level (Q2) can substitute for rating (Q3) in analyses (hypothesis H2). We found a statistically significant positive relationship between the variables ($\rho = 0.84$, $p < 0.001$), and so we can accept H2.

Usability Proxies

To test our hypotheses regarding whether proxies can be used to indirectly determine the usability/acceptability of an app, we analyzed the relationships among the other variables and the acceptance score. Those variables are the time that participants reported using a specific configuration of the app and whether participants found a problem related to the disabled permissions in the app. Those variables correspond to survey questions Q4 and Q1. Figure 6a shows the relationship between the acceptance score (y-axis) and the total usage time (x-axis), and Figure 6b shows the relationship between the acceptance score (y-axis) and whether participants found a problem or not (x-axis). We can see in the figures that participants who reported using the app as much or more than the original app (total usage values 2 and 3) tended to give higher acceptance scores than those who reported using it less (total

Variable		Spearman's Rank Test		Wilcoxon Rank-Sum Test	
Dependent	Independent	ρ	p-value	W	p-value
Acceptance (Q2)	Usage (Q4)	0.47	< 0.001***	993	< 0.001***
Acceptance (Q2)	Problem (Q1)	-0.64	< 0.001***	4290	< 0.001***
Usage (Q4)	Problem (Q1)	-0.49	< 0.001***	3619	< 0.001***

Statistical significance: *** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$

Table 2. Statistical significance results of the relationship between the other usability proxies.

usage value 1). Also, finding a problem affected the scores negatively and caused participants to give lower acceptance scores.

To test the significance of these findings, we ran both the Wilcoxon rank-sum and Spearman's Rank Coefficient tests. The results (Table 2) suggest significant relationships between the variables. The first row indicates a significant positive relationship between the usage time and acceptance score, i.e., the higher the usage time, the higher the acceptance score. The second row shows a negative relationship: whenever participants encountered problems, the acceptance scores declined. Since both relationships are statistically significant ($p < 0.001$), we can accept both H3 and H4.

Moreover, we analyzed the relationship between the usage time and finding a problem as shown in Figure 6c. We found a negative significant relationship between them as shown in the last row of Table 2. This indicates that whenever a problem was encountered, the usage time declined significantly.

Predicting User Ratings

We tested the collaborative filtering algorithm described earlier on 137 ratings from 26 participants who, on average, rated five configurations of the test app. We used the acceptance score, which was recorded on a 7-point Likert scale, as the usability measure. To calculate accuracy, we iteratively considered one of the participants' ratings as unknown and tried to predict it. Then we compared the predicted rating with the known (ground-truth) rating, specifically calculating the mean absolute error (MAE) and root mean square error (RMSE) of the collaborative filtering algorithm.

Before presenting our results, we first provide a concrete example from our dataset. Table 3 shows ratings for one participant ('active user') for whom the rating for node $\{1,2\}$ is being predicted. The 'Actual' column shows the participant's actual ratings for various configurations, including the rating for $\{1,2\}$ in parentheses, which is treated as the ground truth. In the 'Nearest Neighbors' column, we show the ratings of the two closest neighbors of the active user based on their cosine similarity of the rating vectors with the active user. Note that this metric normalizes participants' scores, and so Neighbor 1 is close to the active user despite having generally higher scores. The last column of Table 3 shows the predicted rating for node $\{1, 2\}$. The predicted rating accounts for variations in the average ratings for individual users and in this case results in a prediction of 3.97. Applying the clamping technique, we set the prediction to 3 because 3.97 is greater than the rating value for the child node $\{1\}$ in the lattice structure. In this example clamping yields better results.

Node	Nearest Neighbors		Active User	
	Neighbor 1	Neighbor 2	Actual	Prediction
$\{1\}$	7	3	3	
$\{2\}$	6			
$\{4\}$		1		
$\{1,2\}$	6	4	? (2)	3
$\{1,3\}$	3		3	
$\{2,4\}$		2	2	
$\{3,4\}$	2			
$\{1,3,4\}$		2		
$\{2,3,4\}$			1	

Table 3. Example of the prediction algorithm

# of Nearest Neighbors	Actual		Clamping	
	MAE	RMSE	MAE	RMSE
1	1.402	2.098	1.312	1.974
2	1.375	2.072	1.301	1.987
3	1.317	1.968	1.250	1.896
4	1.301	1.925	1.236	1.845
5	1.285	1.894	1.222	1.820
6	1.273	1.872	1.207	1.794

Table 4. Accuracy of the prediction algorithm

The accuracy of our prediction is shown in Table 4. Based on our data set, we can predict the ratings using up to six nearest neighbors; for more than six neighbors the accuracy does not improve significantly. We can see that the MAE decreases as the number of neighbors increases. We anticipate that if we have larger data set then we can increase the number of nearest neighbors, which may result in better MAE and RMSE.

We compared the accuracy of our predictions with the grand average (mean of all ratings) (MAE=1.63, RMSE=2.13), and item averages (mean of each item's ratings) (MAE=1.397, RMSE=1.925). Our algorithm with the clamping technique performs better than both baselines. Thus we conclude that collaborative filtering presents a viable approach for predicting the acceptability of different permission sets for this app, and is suitable for recommending acceptable configurations to users for various privacy levels (e.g., predicting scores for this app with multiple permissions removed).

DISCUSSION AND LIMITATIONS

We now discuss the limitations of our study and directions for future work.

Sample size. Our study's size (26 participants) was adequate to test various hypotheses with statistical significance but precluded others. For example, we wanted to study whether privacy attitudes affected usability scores and could be used to improve recommendations. We hope to study such effects in a larger study, possibly by recruiting participants online.

Population. We leveraged a local population since our experiment involved an in-person visit to our lab for installation of the ratings app and training, which we thought was prudent to ensure understanding of our compensation scheme. As a re-

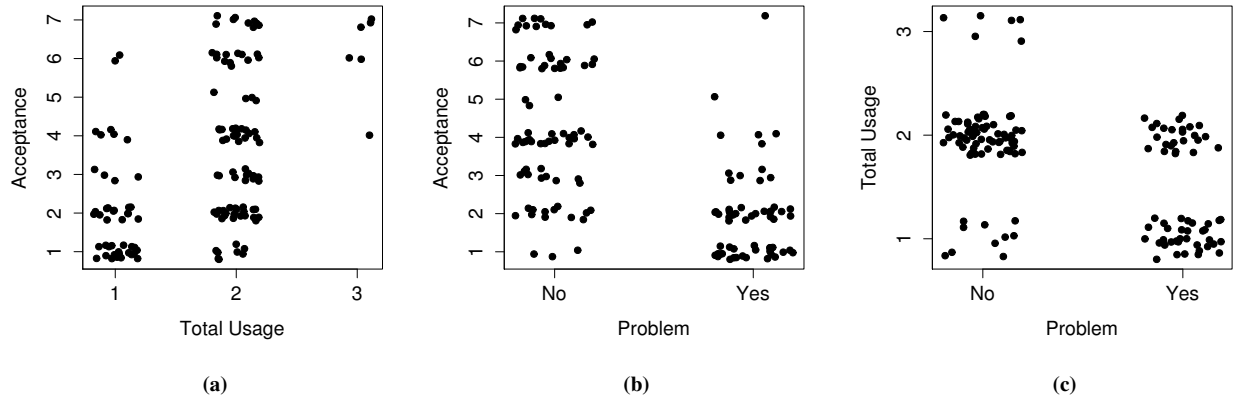


Figure 6. Scatter plots showing the relationships between the other usability proxies a) total usage time and acceptance scores, b) finding a problem in the app and acceptance scores, and c) finding a problem and total usage time.

sult, all participants but one were university students. While our results do not extend to the general population, only 25% of our participants were technical majors and thus our sample was diverse in this sense. Future work may investigate if age is an important factor and, if so, examine crowdsourced exploration based on age groups.

Scalable exploration of different apps. Our study involved only one app (Instagram), which limits the generalizability of our findings. In the future we would like to compare results across various classes of apps such as social media, games, and utilities. While we expect the general pruning strategy to work (based on H1), the lattice may be very large for some apps. To improve scalability, we may monitor which permissions are utilized by the user under normal operation (for example, using the approach by Felt et al. [4]), and then use our crowdsourced approach for subsets relevant to those permissions. Another possibility is to structure the exploration of an app based on knowledge gained from the exploration of another app. If similar users have already been identified, these users can be tasked to explore more diverse configurations, as common configurations are expected to add less information.

Installation process. On Android, users can view the permissions requested by an app during installation. Whether a participant read these permissions was unknown to us, thus possibly introducing a confound where not all participants were aware of the permission set and so were biased differently in their assessments. So, we ordered the survey questions such that participants were first asked to report any problems encountered in the app (Q1) and then informed of the removed permissions and asked to rate the app (Q2). Participants thus reflected upon their actual usage of the app immediately before rating it. Our analysis shows a strong correlation between the problems participants encountered with configurations and their ratings for those configurations, thus indicating their ratings were based largely on their experience.

Incentivizing the crowd. We envision our approach integrated into popular App marketplaces where app ratings can be enhanced to show the top recommended configurations for users. Though recruiting and motivating crowd participation to rate apps with some permissions disabled is not our focus in this paper, we expect that the reciprocal benefits that participation might offer could be a basis for doing so. Popular App marketplaces already demonstrate that users are motivated to rate apps for such reciprocal benefits.

Failure modes. Apps can behave in various ways when permissions are selectively disabled, ranging from crashes to gracefully handling errors. Our approach is primarily relevant to apps that gracefully handle any resulting errors. However, we note that combinations of permissions that render an App useless will be quickly identified by a handful of users and 'pruned' from further exploration. Thus, our approach need not inconvenience a large number of users even for apps that do not handle permission removals gracefully.

Accuracy of the recommendations. Some users may experience problems unrelated to the removed permissions. In our study we observed six such data points; removing these would have improved the MAE of our recommendations by 1.5%. In a real deployment, errors encountered by users could be analyzed for relevance to the particular configuration (e.g., by observing software exceptions). Moreover, like any other recommendation system, malicious users could provide fake ratings in favor of or against specific configurations. For example, Mukherjee et al. [11] analyzed and proposed models for detecting fake product reviews. Although detecting such ratings is out of the scope of this paper, removing those ratings would increase the accuracy of the recommendations.

CONCLUSION

Through a preliminary user study we have shown that it is feasible to apply crowdsourcing as a technique to collectively explore various security configurations of apps and find configurations that make suitable tradeoffs between privacy and usability for a diverse set of users. While others have proposed complementary techniques that rely on automated analyses or seeking opinions of permission settings from the crowd, we

posit that considering opinions based on actual usage of various configurations would facilitate better recommendations to users. However, as the space of all possible permission settings grows exponentially with the number of permissions, it is imperative to follow a thoughtful crowd-management strategy; a goal of this work was to validate our lattice-based strategy for crowd management. Given our promising results, we thus advocate for further, and larger scale, research for crowdsourced exploration of security configurations.

ACKNOWLEDGMENTS

This work was supported by grants 1228364 and 1228471 from NSF. Ismail is funded by the College of Computer and Information Sciences in King Saud University, Saudi Arabia. We thank Steven Armes for earlier work on our software; Shirin Nilizadeh for testing the software; Wesley Beaulieu and Hiroki Naganobori for assistance with statistical analysis; and Norman Su and Kelly Caine for valuable feedback.

REFERENCES

1. Amini, S., Lin, J., Hong, J. I., Lindqvist, J., and Zhang, J. Mobile application evaluation using automation and crowdsourcing. In *Workshop on Privacy Enhancing Tools* (July 2013).
2. Bugiel, S., Davi, L., Dmitrienko, A., Fischer, T., Sadeghi, A.-R., and Shastry, B. Towards taming privilege-escalation attacks on Android. In *19th Network & Distributed System Security Symposium* (Feb. 2012).
3. Ekstrand, M. D., Riedl, J. T., and Konstan, J. A. Collaborative filtering recommender systems. *Foundations and Trends in Human-Computer Interaction 4*, 2 (Feb. 2011), 81–173.
4. Felt, A. P., Chin, E., Hanna, S., Song, D., and Wagner, D. Android permissions demystified. In *18th ACM Conference on Computer and Communications Security* (2011), 627–638.
5. Felt, A. P., Greenwood, K., and Wagner, D. The effectiveness of application permissions. In *2nd USENIX Conference on Web Application Development* (2011), 75–86.
6. Felt, A. P., Ha, E., Egelman, S., Haney, A., Chin, E., and Wagner, D. Android permissions: User attention, comprehension, and behavior. In *8th Symposium on Usable Privacy and Security* (2012), 3:1–3:14.
7. Kelley, P. G., Consolvo, S., Cranor, L. F., Jung, J., Sadeh, N., and Wetherall, D. A conundrum of permissions: Installing applications on an Android smartphone. In *Financial Cryptography and Data Security*. Springer, 2012, 68–79.
8. Kelley, P. G., Cranor, L. F., and Sadeh, N. Privacy as part of the app decision-making process. In *2013 ACM Conference on Human Factors in Computing Systems* (2013), 3393–3402.
9. Kennedy, K., Gustafson, E., and Chen, H. Quantifying the effects of removing permissions from Android applications. In *IEEE Mobile Security Technologies* (2013).
10. Lin, J., Sadeh, N., Amini, S., Lindqvist, J., Hong, J. I., and Zhang, J. Expectation and purpose: Understanding users' mental models of mobile app privacy through crowdsourcing. In *2012 ACM Conference on Ubiquitous Computing* (2012), 501–510.
11. Mukherjee, A., Kumar, A., Liu, B., Wang, J., Hsu, M., Castellanos, M., and Ghosh, R. Spotting opinion spammers using behavioral footprints. In *19th ACM Conference on Knowledge Discovery and Data Mining* (2013), 632–640.
12. Nauman, M., Khan, S., and Zhang, X. Apex: Extending Android permission model and enforcement with user-defined runtime constraints. In *5th ACM Symposium on Information, Computer and Communications Security* (2010), 328–332.
13. Resnick, P., Iacovou, N., Bergstrom, M. S. P., and Riedl, J. T. GroupLens: An open architecture for collaborative filtering of netnews. In *ACM Conference on Computer Supported Collaborative Work* (1994), 175–186.
14. Saltzer, J. H., and Schroeder, M. D. The protection of information in computer systems. *Proceedings of the IEEE 63*, 9 (Sept. 1975).
15. Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. Analysis of recommendation algorithms for e-commerce. In *2nd ACM Conference on Electronic Commerce* (2000), 158–167.
16. Sarwar, B., Karypis, G., Konstan, J., and Riedl, J. Item-based collaborative filtering recommendation algorithms. In *10th International Conference on World Wide Web* (2001), 285–295.

APPENDIX

Here we list the survey questions in the ratings app. In these questions, we referred to a configuration of the test app with some permissions removed as a "version" of the test app, as we feared that "configuration" might confuse participants.

Q1. Were there any features that did not appear to work in this version of the test app? • Yes • No
If (Yes), please provide examples of features that did not work (select all that apply): • Location based features • Camera based features • Microphone based features • The app could not access my contacts • Others:[ ]
Q2. The following features were removed from the test app for this version: • Camera • Location
Would you find it acceptable to use the test app with these features disabled until the next major app update from the test app- 1) Totally unacceptable 2) Unacceptable 3) Slightly unacceptable 4) Neutral 5) Slightly acceptable 6) Acceptable 7) Perfectly Acceptable
Q3. How would you rate this version of the test app compared to the original one? 1) Much worse 2) Somewhat worse 3) About the same 4) Somewhat better 5) Much better
Q4. How often did you use this version of the test app since you installed it on INSTALLATION DATE? a) Not at all b) Less than I normally use the test app c) About as much as I normally use the test app d) More than I normally use the test app