Android User Privacy Preserving Through Crowdsourcing

Bahman Rashidi, *Student Member, IEEE*, Carol Fung, Anh Nguyen, Tam Vu, and Elisa Bertino, *Fellow, IEEE*

***Abstract*— In current Android architecture, users have to decide whether an app is safe to use or not. Expert users can make savvy decisions to avoid unnecessary privacy breach. However, the majority of normal users are not technically capable or do not care to consider privacy implications to make safe decisions. To assist the technically incapable crowd, we propose DroidNet, an Android permission control framework based on crowdsourcing. At its core, DroidNet runs new apps under probation mode without granting their permission requests upfront. It provides recommendations on whether to accept or reject the permission requests based on decisions from peer expert users. To seek expert users, we propose an expertise ranking algorithm using a transitional Bayesian inference model. The recommendation is based on the aggregated expert responses and its confidence level. Our simulation and real user experimental results demonstrate that DroidNet provides accurate recommendations and cover the majority of app requests given a small coverage from a small set of initial experts.**

***Index Terms*— Mobile applications, crowdsourcing, privacy, permission.**

I. Introduction

MOBILE apps have brought tremendous impact to businesses, social, and lifestyle in recent years. Various app markets offer a wide range of apps from entertainment, business, health care and social life. Android app markets, which share the largest user base, have gained a tremendous momentum since its first launch in 2008. According to the report by Android Google Play Store, the number of apps in the store has reached 2.2 million in June 2016, surpassing its major competitor Apple App Store [30]. The rise of Android phones brought the proliferation of Android apps, resulting in an ever-growing application ecosystem [4].

As users rely more on mobile devices and apps, the privacy and security concerns become prominent. Malicious third-party apps not only steal private information, such as contact list, text messages, online accounts, and location from their users, but also cause financial loss to users by making secretive premium-rate phone calls and text messages [29]. At the same time, the rapid growth in the number of apps makes it impractical for app market places, such as Google App Store, to thoroughly verify if an app is malicious or not. As a result, mobile users are left to decide whether an app is safe to use or not. This approach leaves little obstacle for malicious apps to be installed by users.

More specifically, beginning in Android (API level 23), user grants permissions to apps while the apps are running. Users can also manually revoke permissions from any app, even the ones designed for old versions of Android. Unauthorized communications among apps are prohibited. However, such permission control mechanism has been proven to be ineffective in protecting users from malicious apps. A study shows that more than 70% of smartphone apps request to collect data irrelevant to the main function of the app [1]. Among the 1.4 million apps in Google Play, a significant percentage of them have permissions going beyond the apps' intended use. The situation is even worse in the third-party markets which are also available to Android users. In addition, such study shows that only a very small portion (3%) of users pay attention and make correct responses to requests for resource permission at installation, since they tend to rush through to get to use the application. The current Android permission warnings do not help most users make correct security decisions [16].

As pointed out in [15] and [16], the reasons for the ineffectiveness of the current permission control system include: (1) inexperienced users do not realize resource requests are irrelevant and could compromise their privacy, (2) users have the urge to use the app and may be have to give up their privacy in order to use the app. Realizing these shortcomings in the current Android architecture, several efforts have been made to address the problems. Many resource management systems are proposed such as in [22], [25], and [26]. Going down to the system level, L4Android [19] isolates smartphone OS for different usage environments in different virtual machines (VMs). QUIRE [14] provides a set of extensions addressing a form of attack, called *resource confused deputy* attacks, in Android. However, such approaches are not efficient since users are either not paying attention to permissions being requested or not aware of the permissions' implications. Hence, no mechanism that assumes users to have high

Manuscript received February 1, 2017; revised August 27, 2017 and October 13, 2017; accepted October 18, 2017. Date of publication October 26, 2017; date of current version December 19, 2017. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Mauro Barni. *(Corresponding author: Bahman Rashidi.)*

B. Rashidi and C. Fung are with the Department of Computer Science, Virginia Commonwealth University, Richmond, VA 23284 USA (e-mail: rashidib@vcu.edu; cfung@vcu.edu).

A. Nguyen and T. Vu are with the Department of Computer Science, University of Colorado Boulder, Boulder, CO 80309 USA (e-mail: anh.t4.nguyen@ucdenver.edu; tam.vu@ucdenver.edu).

E. Bertino is with the Department of Computer Science, Purdue University, West Lafayette, IN 47907 USA (e-mail: bertino@cs.purdue.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2017.2767019

technical and security knowledge will be usable for a wide audience.

To address this problem, we propose DroidNet, a framework to assist mobile users in controlling their resource usage and privacy through crowdsourcing. First, the framework allows users to use apps without having to grant all permissions. Second, DroidNet allows one to receive help from expert users when permission requests appear. Specifically, DroidNet allows users to install untrusted apps under a *"probation"* mode, while the trusted ones are installed in normal *"trusted"* mode. In probation mode, users make real-time resource granting decisions when apps are running. The framework facilitates a user-help-user environment, where expert users are identified and their decisions are recommended to inexperienced users. To support this user-help-user environment, an effective expert user seeking is the major challenge. DroidNet starts from a small set of trusted expert users (seed users) and propagates the expert evaluation using a transitional Bayesian learning model. We evaluate the effectiveness of the model through simulation and survey data from real users.

The major contributions of this paper include: (1) A comprehensive Android permission control framework to facilitate a user-help-user environment in terms of permission control; (2) A novel transitive Bayesian inference model to propagate expertise rating of users in a network through pairwise similarity among users; (3) A low-risk recommendation algorithm which can help inexperienced users with permission control decision making; (4) A prototype implementation of the system and real user evaluation on the usability of the system.

In the next section, we will provide an in-depth discussion of existing literature in resource management and permission controls for Android applications. We then provide an overview of DroidNet in Section III, followed by the presentation of our algorithms in Section IV. Section V shows the implementation of our system on Android. Section VII shows our evaluation results from our real-life experiments. We study the threats to the system and corresponding mitigation techniques in Section VIII. We conclude the paper by a discussion of future directions.

II. Related Work

Crowdsourcing has been widely applied to address problems ranging from basic to complex in a variety of disciplines, including information systems development, marketing and operationalization [6]. As an application of crowdsourcing, it can be used in the recommendation-based systems [31]. Crowdsourcing acts as a distributed human intelligence agent in a recommendation-based system in which participants? (human individuals) opinions (solutions) are asked and later aggregated to make a recommendation on a decision problem [12].

Exploring user perceptions of privacy on smartphones using crowdsourcing has already been investigated. Agarwal and Hall [5] propose PMP which collects users' privacy protection decisions and analyses them to recommend them to other iOS users. However, their recommendations are based on simple *majority voting* which results in high false recommendation rates.

Liu *et al.* [21] investigated people's privacy preferences by capturing apps logs and analyzing them to identify a small number of profiles that simplify decision makings for mobile users. Profiles were mined from logs by using SVM techniques. However, they do not include users' expertise in their study and this might cause false recommendation.

Lin *et al.* [20] investigated the feasibility of identifying a small set of privacy profiles to help users manage their privacy profiles. Instead of relying on smartphone users' decisions on permission requests, they identified the privacy profiles using Androguard, a static code analysis tool. They analyzed the purpose for which an app requests a permission and identified the permissions that satisfy the least privilege policy. Thus, they can find a set of necessary permissions for apps.

Liu *et al.* [23] proposed PriWe in which they crowdsource users' decisions on permission requests and identify users' expectations. In their work, they focus on finding users with similar responses to permission requests. After finding similar users and applying a recommendation algorithm they identify some privacy profiles and recommend them to those who have similar strategy for responding to permission requests.

Ismail *et al.* [18] propose a crowdsourcing solution to find a minimal set of permissions that will preserve the usability of the app for diverse users. Their approach has a few shortcomings. Repackaging apps for all possible permission combinations is not practical. Also their inability to differentiate between inexperienced and malicious users makes their recommendations of limited quality. Yang *et al.* [32] propose a system to allow users to share their permission reviews with each other. Users leave comments on permissions and the system ranks reviews and recommends top quality reviews to users. Shahriyar proposes Gort [9], an analysis technique that analyzes app behavior while taking into account the context and semantics of the app. Gort uses a three-phase crowd analysis approach, in which crowd workers are asked whether it makes sense for the application to use its requested resources and tasks. App Ops [7], a feature in Android v4.3, allows users to selectively disable permissions for apps on their phones. However, Google removed this feature in their next update, reporting that it was experimental and could cause apps to behave in unexpected ways.

The common feature of those approaches is that they do not consider users' expertise in privacy profiling or permission recommendations. In contrast, considering the fact that most users are inexperienced, we proposed an expertise ranking algorithm to evaluate the expertise level of users for a higher quality recommendations.

As our previous work, our concept paper [27] proposed to utilize expert users' decisions to help inexperienced users on Android permission control. In this paper, we developed an expert seeking and a decision recommendation algorithm based on *transitive Bayesian inference theory*. Furthermore, we implemented the system on Android platform and conducted a set of comprehensive experiments including simulation and real user data to evaluate the performance and usability of the system.

There have been several applications of crowdsourcing in recommendation-based systems in contexts other

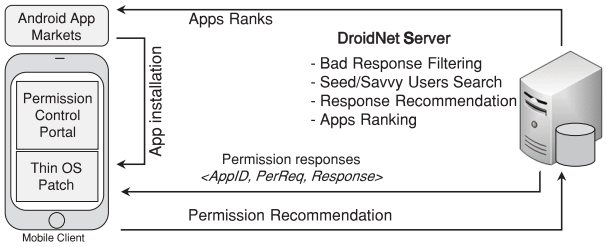


Fig. 1. DroidNet Service Overview.

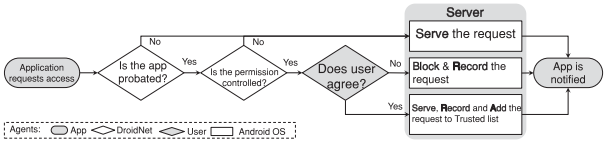


Fig. 2. Permission request flow in DroidNet.

than smaryphone privacy. For example, using crowdsourcing recommendation-based system to learn personal preferences of customers in e-commerce applications [17]. Another implementation of the crowdsourcing in recommendation systems is applying it to tourism recommendation by crowdsourcing social medias [24]. Crowdsourcing recommendation-based systems are also being used to improve the performance of workers at work environments [8], [33]. The role of the recommendation-based system is to collect workers' profiles, explicit worker feedback, user-task interaction and task details and process them in order to make a recommendation on which tasks should be assigned to which worker [8]. Designing a recommendation-based system that achieves high level of accuracy would be a great opportunity for solving issues through a distributed human intelligence system. In order to achieve such high accuracy, designing an effective crowdsourcing method is a fundamental step. Specifically, receiving information and knowledge from a part of crowd that contribute the most (individuals with high expertise related to the crowdsourced problem) helps to increase the quality of recommendations. As we have discussed in this section, similar crowdsourcing and recommendation-based solutions have been used in the smartphone security and privacy area. However, the major difference between our solution and these solutions is that we include users' expertise in our model and recommendations.

III. System Design

Our general approach is to build DroidNet with four functional processes, of which two are on mobile clients and the other two are on remote servers. In particular, DroidNet (1) collects users permission-request responses, (2) analyzes the responses to eliminate untruthful and biased responses, (3) recommends other users low-risk responses to permission requests, and (4) ranks apps based on their security and privacy risk level inferred from users' responses. Fig. 1 shows an overview of DroidNet architecture, which is composed of an application that enables to capture permissions request and receiving recommendations from *DroidNet* service. DroidNet handles the permission requests through the process illustrated in Fig. 2. The differentiating factor of DroidNet is the ability to seek expert user base on a small set of seed users (Section IV). In the rest of this section, we describe four key features of the DroidNet system.

A. Permission Granting Recommendation

To help inexperienced users with their decisions, DroidNet provides recommended responses to users (Fig. 6 (c)). If a user chooses to deny a request, a dummy data or *void* will be returned to the application. For example, a denied GPS location request could be answered with a random location. The user decisions are recorded by the DroidNet client and sent to the DroidNet server for further analysis. In DroidNet, users make decisions twice: (1) selecting the installation mode ("probation","trusted"), (2) responding to the permission requests (once for every selected permissions by user). In a later phase when sufficient data is collected, and a security ranking of the app is available, the DroidNet server can decide whether to pop up permission requests to users or automatically respond them based on prior knowledge. Therefore, DroidNet manages to achieve a balance between the fine-grained control and usability.

B. Bootstrapping the Service

To suggest high-quality recommendations to users, DroidNet starts from a set of trusted *seed expert users* and makes recommendation based on their responses. However, it is impractical to have such expert users providing responses to all available apps on the market due to the extremely large number of these apps. To address this scalability challenge, we propose a spanning algorithm that searches for *external expert users* based on the similarity of their responses to DroidNet internal experts set, in combination with the user's accumulative reputation. Using this algorithm, if there is more overlap (requests that users have responded to) between the seed experts and users, it helps to estimate the expertise of users more accurately. To achieve such goal, the seed expert cover the most installed apps. This can help to generate more overlap among users and seed experts. The other reason for covering most installed apps by seed experts is that the number Android apps' downloads on markets follow the *long-tail* distribution [2], [3]. A long-tail distribution is when the portion of the distribution having a large number of occurrences far from the *"head"* or central part of the distribution [13]. As reported [2], [3], there is only 0.001 of apps on Android markets with number of downloads more than 1M. Fig. 10(a) shows an example of a long-tail distribution. We explain the process in details in the evaluation section.

IV. Expert Users Seeking

The key challenge of DroidNet is to seek experts from the regular users in the DroidNet. In DroidNet users' responses to permission requests are recorded by a central server and the responses from expert users are used to generate recommendations to help inexperienced users make low-risk decisions. DroidNet starts from a small set of trusted seed expert users,

TABLE I
NOTATIONS

Notation	Description
$\mathcal{U}$	$\{U_1, ..., U_n\}$: Set of n DroidNet users in the system.
s	The seed user.
$\mathcal{R}_i$	The set of requests responded by user i in the past.
p_i	The true expertise level of user i.
R_i, C_i	The expertise rating and rating confidence of user i.
$(\alpha_{ij}, \beta_{ij})$	The similarity tuple between user i and j.
(α_i, β_i)	The expertise level distribution parameters for user i.

and propagate the expertise evaluation based on similarity among users using a transitive Bayesian inference model [28]. This section describes the model in more detail.

A. Assumptions and Notations

DroidNet can be seen as a network $\mathcal{G} = \{s \cup \mathcal{U}, \mathcal{E}\}$, which consists of a *seed expert* s, a set of n regular users $\mathcal{U} = \{U_1, U_2, \ldots, U_n\}$, and a set of edges $\mathcal{E} = \{e_{ij} | \mathcal{R}_i \cap \mathcal{R}_j \geqslant \theta, \forall i, j\}$, where $\mathcal{R}_i(j)$ denotes the set of permission requests answered by user $i(j)$. Edge $e_{ij} \in \mathcal{E}$ denotes the set of permission requests to which both users have answered. Users are connected if the number of commonly answered requests exceeds threshold θ.

The *seed expert* (SE) is one or a set of trusted expert users who are be employed by a DroidNet facilitator to provide *correct* responses to permission requests. It is worth noting that the seed experts follow the *principle of least privilege*, where a minimal set of permissions that are necessary for apps' legitimate purposes are defined to determine the correct responses for permission requests. This way, their responses do not depend on user preferences or context. However, due to the high cost of human labor, the seed expert can only cover limited number of applications. Therefore, identifying expert users from regular users can expand the coverage of apps that can benefit from DroidNet recommendations.

Let $\mathcal{R}_s$ denote the set of requests covered by seed experts. Then the common set of requests answered by both the seed user and user i can be written as $\mathcal{R}_{si} = \mathcal{R}_s \cap \mathcal{R}_i$. Table I lists the notations we use in this paper.

B. The Users Expertise Rating Problem

Before getting into more details, we show a general view of the users network. Fig. 3(a) presents a comprehensive view of DroidNet's user network. In this network we can see different types of users and communities. Fig. 3(b1),(b2) show the User-Seed and User-User connections.

The *expertise level* of a user i, denoted by $p_i \in [0, 1]$, is the likelihood that the user makes correct permission granting decisions. Given the set of responses that user i has given to permission requests and their corresponding ground truth, a Bayesian inference model can be used to estimate p_i.

Definition 1 (Expertise Rating and Rating Confidence): Assume that the likelihood that a user i makes correct decision (p_i) satisfies a distribution Y_i with pdf $f_i(x)$.

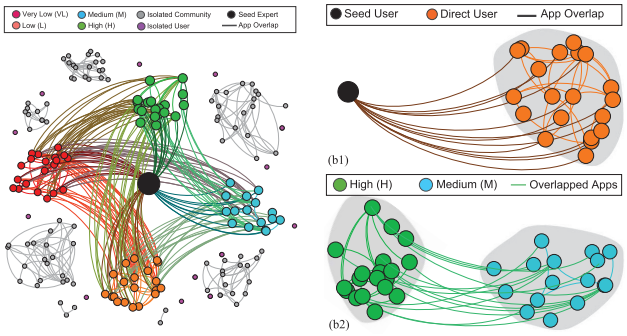


Fig. 3. DroidNet user connectivity network: (a) overall view of the network; (b) users' connectivity (User-Seed and User-User).

Then we define the estimated expertise level of the user to be:

$$R_i = \mathbb{E}[Y_i] = \int_{x=0}^{1} x f_i(x) dx,$$

The confidence level of the estimation is:

$$C_i = 1 - \theta\delta[Y_i] = 1 - \theta\Big(\int_{x=0}^{1} (x - R_i)^2 f_i(x) dx\Big)^{1/2}$$

where θ is the normalization factor. Therefore, the expertise seeking problem can be described as follows:

Problem 2 (Expertise Rating Problem): Given a seed user s, a set of users $\mathcal{U} = \{U_1, \ldots, U_n\}$, and a DroidNet graph $\mathcal{G} = \{\mathcal{U} \cup s, \mathcal{E}\}$. The expertise rating problem is to find the posterior distributions of all p_i, given their past history of responses to permission requests.

Before presenting the solution, we first define the concept of similarity and then discuss a special case where a user is connected to a seed expert only.

Definition 3 (Similarity of Two Users): Let i and j be two users who have responded to a common set of permission requests $\mathcal{R}_{ij}$, then we define the similarity between i and j as the tuple $(\alpha_{ij}, \beta_{ij})$, where α_{ij} and β_{ij} denote the accumulated number of consistent responses and inconsistent responses to those common requests, respectively.

Let $\{x_k \in \{0, 1\} | 1 \leq k \leq n\}$ denote a sequence of n observations in history, where $x_k = 1$ means that the two users provided consistent responses at the kth overlapped request, and vice versa. The similarity tuple can be computed as follows:

$$\begin{aligned} \alpha_{ij}^{(n)} &= \sum_{k=1}^{n} q^{n-k} x_k + q^n C_0 \\ &= x_n + q x_{n-1} + \ldots + q^{n-1} x_1 + q^n C_0 \end{aligned} \tag{1}$$

$$\begin{aligned} \beta_{ij}^{(n)} &= \sum_{k=1}^{n} q^{n-k} (1 - x_k) + q^n C_0 \\ &= (1 - x_n) + q(1 - x_{n-1}) + \ldots + q^{n-1}(1 - x_1) + q^n C_0 \end{aligned} \tag{2}$$

Where C_0 is a constant weighting the initial belief; $q \in [0, 1]$ is the remembering factor which is used to discount the influence from past experience and therefore emphasizes the importance of more recent observations.

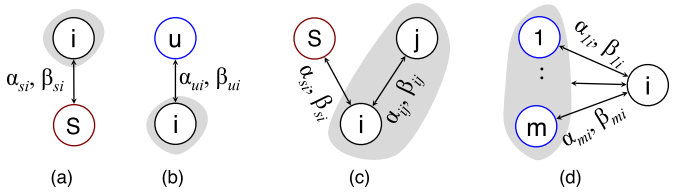


Fig. 4. The illustration of four cases of DroidNet graphs. (a) A user is connected directly to a seed user; (b) a user is connected to a non-seed user; (c) a multi-hop rating propagation case; (d) a multi-path rating aggregation case.

C. Users Connected to the Seed Expert

We start with the case that a user i who has a common set of responded requests with the seed expert (see Fig. 4(a)). In such case, our approach is to compute the similarity tuple $(\alpha_{si}, \beta_{si})$ between the user and the seed, and then the distribution of p_i based on the observations.

We have the following Lemma:

Lemma 4: Let i be a user i that has only one seed expert neighbor in the DroidNet graph. Let $(\alpha_{si}, \beta_{si})$ be the similarity tuple of i and the seed expert. Then the rating of the user can be estimated as follows:

$$R_i = \frac{\alpha_{si}}{\alpha_{si} + \beta_{si}} \tag{3}$$

$$C_i = 1 - \sqrt{\frac{12\alpha_{si}\beta_{si}}{(\alpha_{si} + \beta_{si})^2(\alpha_{si} + \beta_{si} + 1)}} \tag{4}$$

Proof: Since the seed expert's advise is assumed correct, α and β are indeed the number of correct and incorrect responses that the user answered in the past. Let a random variable $X \in \{0, 1\}$ denote whether a user answers the permission requests correctly or not. $X = 1$ indicates that user responds to a request correctly, vice versa. Therefore, we have $p = \mathbb{P}(X = 1)$. Given a sequence of observations on X, a beta distribution can be used to model the distribution of p.

In Bayesian inference theory, posterior probabilities of Bernoulli variable given a sequence of observed outcomes of the random event can be represented by a Beta distributions. The Beta-family of probability density functions is a continuous family of functions indexed by the two parameters α and β, where they represent the accumulative observation of occurrence of outcome 1 and outcome 0, respectively. The beta PDF distribution can be written as:

$$f(p|\alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} p^{\alpha-1}(1 - p)^{\beta-1} \tag{5}$$

The above can also be written as,

$$p \sim \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} y^{\alpha-1}(1 - y)^{\beta-1} \tag{6}$$

According to Definition 1, we have Equation (3) and (4). □

D. Users Connected to a Regular User

Due to the limited coverage of the seed user, there may be many users who do not have direct overlap with the seed user (see Fig. 4(b)). To rate users who are connected only to a regular user with known expert rating, we can use the following theorem:

Theorem 5: Let i be a user connected to a user u with known expertise level $p_u > \frac{1}{2}$ in the DroidNet graph; let $(\alpha_{ui}, \beta_{ui})$ be the similarity tuple of i and u, where $\alpha_{ui} \geq \beta_{ui}$. Then p_i satisfies a Beta distribution: $p_i \sim Beta(\alpha_i, \beta_i)$, where

$$\alpha_i = \frac{\alpha_{ui}\, p_u + \beta_{ui}(p_u - 1)}{2p_u - 1} \tag{7}$$

$$\beta_i = \frac{\alpha_{ui}(p_u - 1) + \beta_{ui}\, p_u}{2p_u - 1} \tag{8}$$

Proof: Let random variables $X_i \in \{0, 1\}$ and $X_u \in \{0, 1\}$ denote a random event that user i and u respond to permission requests correctly or not. $X_i(X_u) = 1$ means that user $i(u)$ responds to a permission request correctly. Therefore, we have $p_i = \mathbb{P}(X_i = 1)$ and $p_u = \mathbb{P}(X_u = 1)$

Using Bayes theory, the probability that a consistent response being a correct response is formulated as follows:

$$\begin{aligned}
&\mathbb{P}(X_i = 1|X_i = X_u) \\
&= \frac{\mathbb{P}(X_i = X_u|X_i = 1)\mathbb{P}(X_i = 1)}{\mathbb{P}(X_i = X_u)} \\
&= \frac{\mathbb{P}(X_u = 1|X_i = 1)\mathbb{P}(X_i = 1)}{\mathbb{P}(X_i = 1, X_u = 1) + \mathbb{P}(X_i = 0, X_u = 0)} \\
&= \frac{\mathbb{P}(X_u = 1)\mathbb{P}(X_i = 1)}{\mathbb{P}(X_i = 1)\mathbb{P}(X_u = 1) + \mathbb{P}(X_i = 0)\mathbb{P}(X_u = 0)} \\
&= \frac{p_i\, p_u}{p_i\, p_u + (1 - p_i)(1 - p_u)}
\end{aligned} \tag{9}$$

Similarly, the probability that an inconsistent response being a correct response is formulated as follows:

$$\begin{aligned}
\mathbb{P}(X_i = 1|X_i \neq X_u) &= \frac{\mathbb{P}(X_i = X_u|X_i \neq 1)\mathbb{P}(X_i = 1)}{\mathbb{P}(X_i \neq X_u)} \\
&= \frac{p_i(1 - p_u)}{p_i(1 - p_u) + (1 - p_i)\, p_u}
\end{aligned} \tag{10}$$

Note that α_i and β_i denote the cumulative observations that user i responds correctly and incorrectly respectively. Then α_i and β_i can be obtained indirectly from α_{ui} and β_{ui} from the formula below,

$$\begin{aligned}
\alpha_i &= \alpha_{ui}\mathbb{P}(X_i = 1|X_i = X_u) + \beta_{ui}\mathbb{P}(X_i = 1|X_i \neq X_u) \\
\beta_i &= \alpha_{ui}\mathbb{P}(X_i = 0|X_i = X_u) + \beta_{ui}\mathbb{P}(X_i = 0|X_i \neq X_u)
\end{aligned}$$

The above equation set can be transformed into:

$$\alpha_i = \frac{\alpha_{ui}\, p_i\, p_u}{p_i\, p_u + (1 - p_i)(1 - p_u)} + \frac{\beta_{ui}\, p_i(1 - p_u)}{p_i(1 - p_u) + (1 - p_i)\, p_u} \tag{11}$$

$$\beta_i = \frac{\alpha_{ui}(1 - p_i)(1 - p_u)}{p_i\, p_u + (1 - p_i)(1 - p_u)} + \frac{\beta_{ui}\, p_u(1 - p_i)}{p_i(1 - p_u) + (1 - p_i)\, p_u} \tag{12}$$

Note that the estimated expertise level of user i can be written as $R_i = \alpha_i/(\alpha_i + \beta_i)$. However, the actual expertise level p_i of user i is unknown. An iterative method can be used to iteratively update Equations (11) and Equation (12) starting from $R_i^{(0)} = \frac{1}{2}$ and at each round t replaces p_i with the last round expertise level $R_i^{(t-1)}$. The process stops when $R_i^{(t)}$ converges.

Alternatively we can solve Equation set (11) and (12) by replacing p_i with $\alpha_i/(\alpha_i+\beta_i)$. Then we get (7) and (8). □

E. Multi-Hop User Rating Propagation

Since not all users are connected to the seed user, a rating propagation model is called upon to rate users who are indirectly connected to the seed. As shown in Fig. 4(c), user i has overlap with the seed user, so it can be ranked through our Bayesian ranking algorithm described in Lemma 4. User j only has overlap with user i, so it can be ranked based on its similarity to user i. However, Theorem 5 only works when the expertise of user i is known. Therefore, here we use an iterative method to update the rating of all regular users in DroidNet.

Corollary 5.1: Let i be a regular user directly connected to a set of users $\mathcal{N}_i$. The ratings of the neighbors at round t are (α_i^t, β_i^t), then the rating tuple $(\alpha_i^{(t+1)}, \beta_i^{(t+1)})$ of user i at time $t+1$, can be computed as follows:

$$\begin{aligned}
\alpha_i^{(0)} &= \beta_i^{(0)} = 1, \quad \forall i, \quad s.t.\ U_i \in \mathcal{U} \\
\alpha_i^{(t+1)} &= \sum_{k\in\mathcal{N}_i}\Big(\frac{\alpha_{ik}\alpha_i^{(t)}\alpha_k^{(t)}}{\alpha_i^{(t)}\alpha_k^{(t)}+\beta_i^{(t)}\beta_k^{(t)}}+\frac{\beta_{ik}\alpha_i^{(t)}\beta_k^{(t)}}{\alpha_i^{(t)}\beta_k^{(t)}+\alpha_k^{(t)}\beta_i^{(t)}}\Big) \\
\beta_i^{(t+1)} &= \sum_{k\in\mathcal{N}_i}\Big(\frac{\alpha_{ik}\beta_i^{(t)}\beta_k^{(t)}}{\alpha_i^{(t)}\alpha_k^{(t)}+\beta_i^{(t)}\beta_k^{(t)}}+\frac{\beta_{ik}\alpha_k^{(t)}\beta_i^{(t)}}{\alpha_i^{(t)}\beta_k^{(t)}+\alpha_k^{(t)}\beta_i^{(t)}}\Big)
\end{aligned} \tag{13}$$

Proof: From Equation (11) and (12) we learn that the rating of a node can be computed using the similarity with a source of known rating. We use (α_i^k, β_i^k) denote the transformed observation on user i passed by user k, then we have:

$$\begin{aligned}
\alpha_i^k &= \frac{\alpha_{ki}p_ip_k}{p_ip_k+(1-p_i)(1-p_k)}+\frac{\beta_{ki}p_i(1-p_k)}{p_i(1-p_k)+(1-p_i)p_k} \\
\beta_i^k &= \frac{\alpha_{ki}(1-p_i)(1-p_k)}{p_ip_k+(1-p_i)(1-p_k)}+\frac{\beta_{ki}p_k(1-p_i)}{p_i(1-p_k)+(1-p_i)p_k}
\end{aligned}$$

By replacing p_i with $\frac{\alpha_i}{\alpha_i+\beta_i}$ and p_k with $\frac{\alpha_k}{\alpha_k+\beta_k}$, we have:

$$\begin{aligned}
\alpha_i^k &= \alpha_{ki}\frac{\alpha_k\alpha_i}{\alpha_k\alpha_i+\beta_k\beta_i}+\beta_{ki}\frac{\beta_k\alpha_i}{\beta_k\alpha_i+\alpha_k\beta_i}, \forall k\in\{1,2,\ldots,m\} \\
\beta_i^k &= \alpha_{ki}\frac{\beta_k\beta_i}{\alpha_k\alpha_i+\beta_k\beta_i}+\beta_{ki}\frac{\alpha_k\beta_i}{\beta_k\alpha_i+\alpha_k\beta_i}, \forall k\in\{1,2,\ldots,m\}
\end{aligned}$$

In Bayesian inference theory, the observations on one variable can be cumulated through simple summation on all observations, given that they are observed independently. In this case the rating of a user can be represented by the total number of positive and negative observations observed by connected users on different paths. Given that α_i and β_i represent the cumulative positive/negative observations on user i, we have:

$$\begin{aligned}
\alpha_i &= \alpha_i^1+\ldots+\alpha_i^m = \sum_{k=1}^m \alpha_i^k \\
\beta_i &= \beta_i^1+\ldots+\beta_i^m = \sum_{k=1}^m \beta_i^k
\end{aligned} \tag{14}$$

□

F. Multi-Path User Rating Aggregation

A user may have overlap with multiple other users. As shown in Fig. 4(d), user i is connected to m other users. The overlap with multiple users can be seen as observations from multiple sources and those observations can be aggregated to generate a more accurate ranking of user i.

Corollary 5.2: Let i be a user who has overlap with a set of users $\mathcal{M}=\{U_1,U_2,\ldots,U_m\}$ with corresponding similarity tuples $\mathcal{S}=\{(\alpha_{1i},\beta_{1i}),\ldots,(\alpha_{mi},\beta_{mi})\}$. Then we have:

$$\begin{aligned}
\alpha_i &= \alpha_i^1+\ldots+\alpha_i^m = \sum_{k=1}^m \alpha_i^k \\
\beta_i &= \beta_i^1+\ldots+\beta_i^m = \sum_{k=1}^m \beta_i^k
\end{aligned} \tag{15}$$

where,

$$\begin{aligned}
\alpha_i^k &= \alpha_{ki}\frac{\alpha_k\alpha_i}{\alpha_k\alpha_i+\beta_k\beta_i}+\beta_{ki}\frac{\beta_k\alpha_i}{\beta_k\alpha_i+\alpha_k\beta_i}, \forall k\in\{1,2,\ldots,m\} \\
\beta_i^k &= \alpha_{ki}\frac{\beta_k\beta_i}{\alpha_k\alpha_i+\beta_k\beta_i}+\beta_{ki}\frac{\alpha_k\beta_i}{\beta_k\alpha_i+\alpha_k\beta_i}, \forall i\in\{1,2,\ldots,m\}
\end{aligned}$$

Proof: This results is derived from Corollary 5.1 by iteratively computing α_i and β_i on node i in a graph starting from initial setting $\alpha_i^{(0)}=1$ and $\beta_i^{(0)}=1$. □

Algorithm 1 Rate All Regular Users

```
Compute expertise rating of all regular users in DroidNet
Notations:
R(U): the current rating of all users
R̂(U): the last round rating of all users
s: the seed expert
U_i: the i_th user
G = (V, E): the generated graph of users and overlaps
RU: the set of rated users
QU: the queue of users to be rated
//parameters initialization
set R(s) = 1 and R(U) = 0.5
while (Distance(R(U), R̂(U)) > ε) do
    RU ← s
    QU ← findNeighbors(s)
    R̂(U) ← R(U)
    while (u ← remove(QU) is not null) do
        //Users rating using Corollary 5.2
        R(u) ← computeRating(u)
        RU ← RU ∪ u
        N = findNeighborsNotInRUorQU(u, G)
        push(N, QU)
    end while
end while
```

Our approach to determine the order of user rating is to start from the direct neighbors of the seed user, and then we expand the list by looking for the next hop users, and so on. An iterative algorithm is described in Algorithm 1 which rates all regular users in DroidNet. The iteration stops when the difference between two rounds of ratings are sufficiently close.

G. Recommendation Algorithm

After rating users in the network, the next phase is to generate recommendations based on responses from expert users. We propose a weighted voting method to handle the decision making. The voting process is divided into three phases: qualification, voting, and decision. The algorithm is described in Algorithm 2. In the qualification phase, only responses from qualified users are included into the voting process. Initially the ballot count for reception and rejection decisions are equally initialized to D_0. For each qualified voter, the weight of the cast ballot is the rating score of the voter. After the voting process finishes, the average ballot score is used for a final decision. If the average ballot score exceeds a decision threshold (τ_d), then corresponding recommendations are made. Otherwise, no recommendation is made.

Algorithm 2 Weighted Voting for Recommendation

Notations :
$R(u), C(u)$:the rating score and rating confidence of user u
$x(u)$:the response to permission request from user u
τ_e, τ_c :the minimum rating score and rating confidence to be considered as an expert user
τ_d :the recommendation threshold
a, b :the cumulative ballots for *yes* or *no* decision
D_0 :the initial ballot count for both decisions
$a = b = D_0$
//Users filtering and ballots casting
for each user u who responded to the request **do**
  if $R(u) > \tau_e$ and $C(u) > \tau_c$ **then**
    if $x(u) = 1$ **then**
      $a+ = R(u)$
    else
      $b+ = R(u)$
    end if
  end if
end for
//decision making based on final ballots count
if $\frac{a}{a+b} > \tau_d$ **then**
  Recommend to *accept* the request with confidence $\frac{a}{a+b} - \tau_d$
else if $\frac{a}{a+b} < 1 - \tau_d$ **then**
  Recommend to *reject* the request with confidence $1 - \frac{a}{a+b} - \tau_d$
else
  No recommendation
end if

V. Implementation

The goal of DroidNet is to provide a platform for users to grant permissions to apps based on recommendations from expert users. To prove the concept feasibility, we implemented a prototype of DroidNet. More specifically, we modify the permission management component of the Android operating system, and developed an Android application allowing users to monitor and manage resource access permissions at fine-grain level. Fig. 5 shows DroidNet's implementation architecture. DroidNet is installed by applying a software patch which includes modifications on the Android operating system level and a pre-installed app `DroidNet.apk` on the application level.

A. Permission Control User Interaction

DroidNet users have an option to install apps under a *probation mode*. We use the app "Telegram" (a popular chat application) as an example. The first screenshot (Fig. 6(a)) displays two options when installing the app on the smartphone, e.g. *probation mode* and *trusted mode*. If the user selects the probation mode, then the application will be added to a list of monitored apps on the phone. On the other hand, if the user selects the trusted mode, then the application will be installed with all requested permissions granted.

For each installed app, users can use the pre-installed DroidNet application to view a list of apps which are under the probation mode. If the user clicks on an app in the list, a set of requested resources is displayed (see Fig. 6(b)) where checked resources are monitored. By default all sensitive resources are monitored; however users can change this default.

If an app is installed under the probation mode, whenever the app requests to access to a resource under monitoring, the user is informed by a pop-up (Fig. 6(c)). In addition, DroidNet recommends a decision to users with a confidence level. If the user chooses to follow the recommendation, the request of the application will be served; otherwise the request will be blocked.

B. Android Framework Modification

To implement a real-time resource permission control, DroidNet monitors all resource access requests (system calls) at runtime. We modified a few components and methods in Android framework to meet our goal.

1) App Installation Mode: To allow users to have the option to install under probation or trusted mode, we modified the *Package Manager Service*, which plays the main role in the installation of apps and their requested permissions management. Installation is managed by the *PackageInstaller* activity and when an application installation is completed, a notification is sent to *InstallAppProgress.java*, which is the place we added a post install prompt to ask users if they would like to put application on probation mode. If a user selects the trusted mode installation then the app would not be managed by DroidNet, and no information will be recorded about the application. If the user selects the probation mode, DroidNet records app's UID and the set of requested permissions by the probated app in the *Probated Apps Repository* (PAR) and *Request/Recommendation Repository* (RRR) repositories. Note that communication is supported through by using these repositories that all layers (framework and application) read and write from.

2) System Calls Monitoring and Permission Enforcement: Our implementation is designed to be extensible and generic. While our implementation requires multiple changes in one

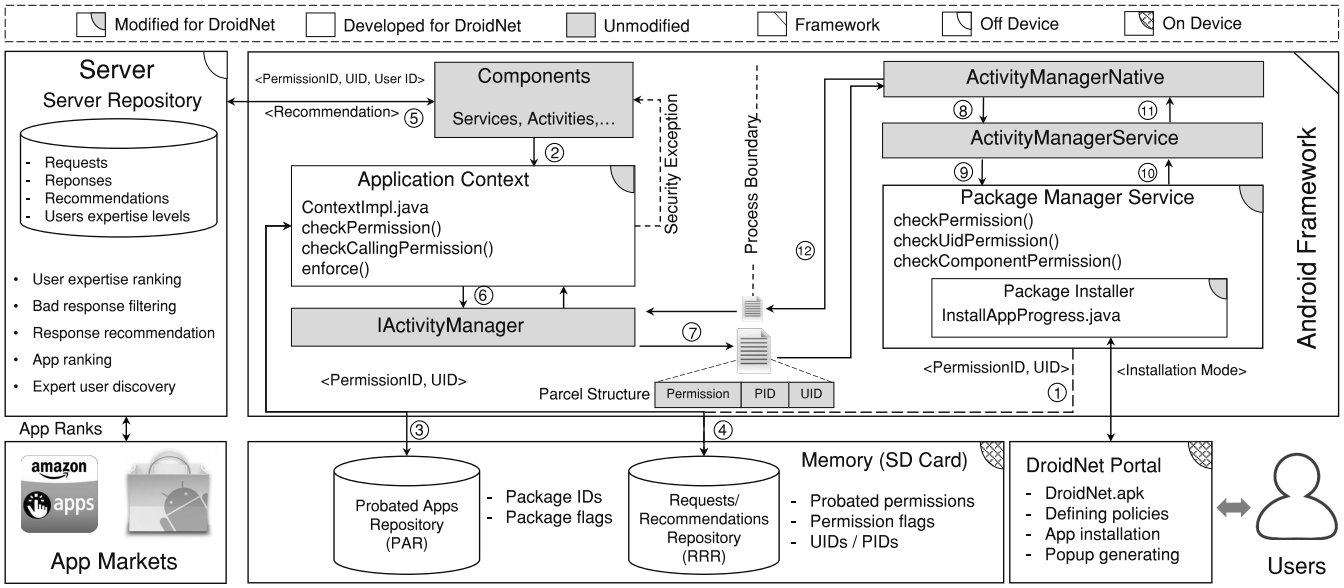
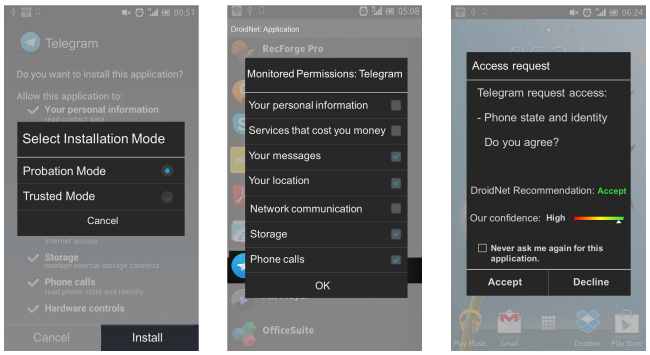


Fig. 5. DroidNet implementation architecture overview.



(a) (b) (c)

Fig. 6. An example of DroidNet on Telegram app: (a) probation and trusted installation modes; (b) users pick which critical resources to be monitored; (c) pop-up for permission granting with suggestion from DroidNet and its confidence.

place, it does not require modifications on every permission request handler, as it was the case on some previous works, such as in MockDroid [11]. The modification is presented in the form of an framework patch, which can be executed from a user's space, making this technique easier to adopt. In order to design an extensible and central permission enforcing point, we modified methods such as `enforcePermission`, `checkPermission`, and `enforceCallingPermission` of *ContextImpl.java* class of the *context* component of Android. These methods are called whenever an application seeks to use some permissions that are not hardware related. When the methods are called, it is passed a UID and a permission name. We first check to see if the UID is a system call. If yes then we check the repository to see if the UID is present, and if it is, what value the flag associated with the passed in permission has. Algorithm 3 shows the flow of permission enforcement for incoming permission requests.

C. DroidNet Recommendation Server

Recording the users' responses and providing decision recommendations to users are essential to DroidNet. For this purpose we maintain a remote server to record the responses on an online server and also compute recommendations according to the recorded responses from users. The DroidNet clients request recommendations from the server when needed.

VI. Performance Evaluation

For a comprehensive evaluation of the DroidNet system, we use simulation to evaluate the performance of the expert rating and recommendation algorithms.

A. Simulation Setup

As a proof of concept, we created a set of DroidNet users' profiles consisting of four different levels of expertise. The expertise level we refer here is the probability that a user responds to permission requests correctly (a.k.a. consistent with the correct responses). User profiles consist of users with a high (H) expertise (0.9), medium (M) expertise (0.7), low (L) expertise (0.5), and the remaining are users with a very low (VL) expertise (0.1). Note that VL is considered to be malicious since their responses are misleading most of the time. In order to measure the effectiveness of the DroidNet expertise rating, we use a few study cases of multi-hop and multi-path propagation. In all experiments we set $q = 1\%$.

Our simulation environment is C++ on a Windows machine with 3.6Ghz Intel Core i7 and 16GB RAM. All results are based on an average of 500 repeated runs with different random generator seeds.

B. Expertise Rating and Confidence Level

To evaluate the effectiveness of the rating and recommendation model, we start from 4 study cases on a set of nodes with

Algorithm 3 Permission Enforcing Flow

This algorithm is to decide whether to grant a requested permission to app or deny it
Notations :
PAR :Probated Apps Repository
RRR :Request/Recommendation Repository
$flag$:denotes that permission is probated
uid :Package identifier
p :Permission name (identifier)
r :user's response for a permission request
//initialize voting parameters
while (there is an incoming permission request) **do**
  Fetch UID's info from PAR
  if $uid \notin PAR$ **then**
    //grant the requested permission
  else
    Fetch apps' probated Ps from RRR
    if p's $flag = True$ **then**
      //grant the requested permission
    else
      Prompting user through a popup
      if $r = True$ **then**
        //grant the requested permission and record the user's response
      else
        //deny the request and record the user's response
      end if
    end if
  end if
end while

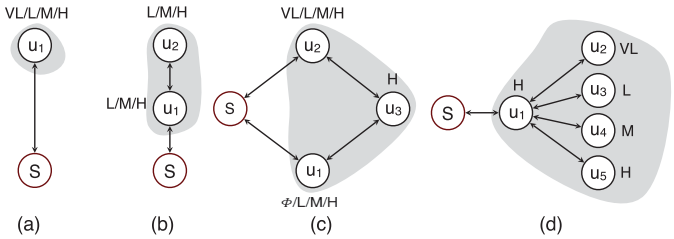


Fig. 7. The illustration of four small user profiles designed for our evaluation: (a) the user is connected directly to a seed user; (b) the user is far from seed by distance 1 intermediate user; (c) the multi-path rating propagation case; (d) a multi-path rating case, designed for α and β calculation convergence.

designed configuration. The average number of permission requests per app is set to 5 and the maximum number of requests is 500. Fig. 7 shows the four case studies and their configurations.

We start from a simple case study in which a user is connected to a seed expert only (Fig. 7(a)), and study the expertise ratings and rating confidence when the user is initialized with H, M, L and VL expertise ratings, respectively. Fig. 8(a) shows the estimated expertise rating for all four types of user's expertise. We can see that when the number of overlapped requests increases, the estimated expertise ratings approach to their true expertise levels. Fig. 8(b) shows the corresponding confidence levels of estimation. The confidence

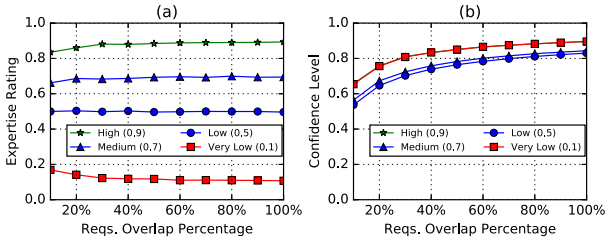


Fig. 8. Calculated user expertise and confidence level: (a) expertise level of user with different initial expertise level; (b) computed confidence level of user with different initial expertise level.

level also increases with the number of overlapped apps. From these results, we can see that DroidNet can have high quality users' expertise rating when the user has sufficient requests overlapping with the seed expert.

In the second case study (Fig. 7(b)), we investigate the influence of intermediate users on the expertise rating propagation. We set user 1 with L, M and H expertise and user 2 with H, M, and L expertises. Fig. 9(a)(b) shows that the expertise rating of user 2 is influenced by the expertise level of user 1. The higher expertise of the intermediate node, the closer user 2's is rated to its actual expertise level. We call a high expertise node has a *high rating conductivity*. We also conclude that through multi-hop rating propagation, we can find expert users who do not have direct overlap with the seed expert.

In the third case study (Fig. 7(c)), we show the expertise rating of user 3 with two intermediate users between user 3 and the seed expert. We vary the type of user 1 to be void (non-existing), H, M, and L expertise and vary the type of user 2 to be H, M, L, and VL. Fig. 9(b) shows that the conductivity of rating is high if one of the paths has high conductivity node.

Fig. 9(c) shows the expertise rating of five users (with expertise $0.1, 0.5, 0.7$, and 0.9) for the case study shown in Fig. 7(d). As we described in Algorithm 1, we continue updating the expertise rating parameters (α and β) of a user until they converge to a stable value. In this experiment, we show the convergence speed of different types of users through iterating α and β calculation process 10 times start from 1 iteration to 10 iterations. From these results we can see that after 10 iterations all ratings converge to stable values, while the user directly connected to the seed expert achieves stableness after one computation cycle.

In the next experiment we test our technique on a medium size network with 400 users and 250 apps in total (Fig. 3(a)). We set up 100 user for each type (VL, L, M and H). Users choose to install 20 apps out of 250 apps randomly. Fig. 9(d) presents the distribution of the final expertise ratings for all 400 users marked by different colors. We can see that the estimated expertise ratings are clustered around their actual expertise levels; however false positives and false negatives exist.

C. Quality of DroidNet Recommendations

Making accurate permission granting recommendations is the main purpose of DroidNet. We thus evaluate the quality

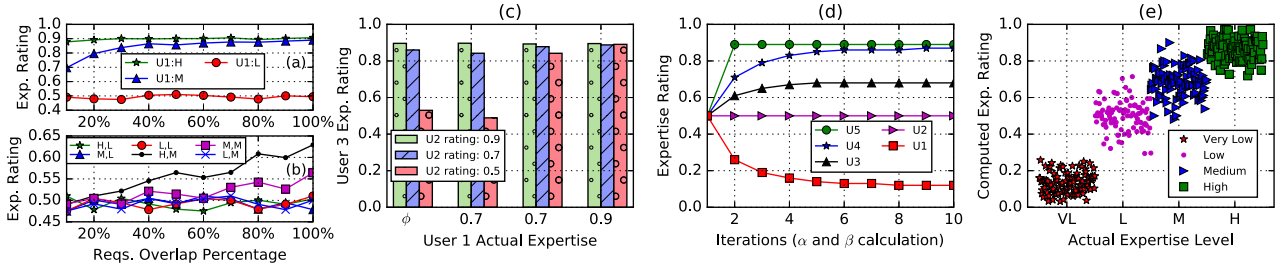


Fig. 9. Influence of neighbors on expertise rating: (a)(b) expertise rating of a user with only one user in its locality and different expertise ratings (U1,U2); (c) expertise rating of a user with two users in its locality and different expertise levels; (d) expertise rating of users for different number of α and β calculation iterations; (e) expertise rating distribution after rating users with different actual expertise rating.

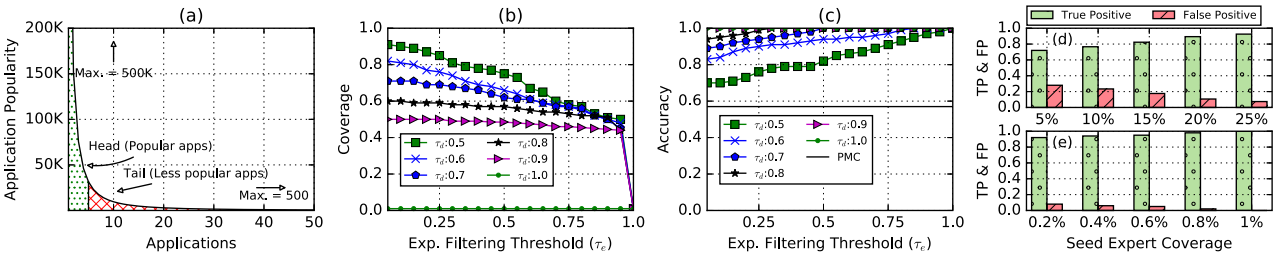


Fig. 10. Coverage and accuracy of rating and recommendation: (a) generated dataset based on teh Long-tail distribution; (b) percentage of requests that DroidNet makes recommendation for; (c) percentage correct recommendations that DroidNet makes; (d)(e) accuracy of generated recommendations and seed expert coverage relation.

of DroidNet recommendations using two metrics: *coverage* and *accuracy*. Coverage is the percentage of the requests for which DroidNet can offer recommendation to users, while accuracy is the percentage of correct recommendation that DroidNet makes. Note that if a request is covered by a seed expert, DroidNet always recommends the response from the seed expert.

In order to conduct the simulation close to the real-life scenario as much as possible, we decided to build a network of users and apps proportionally close to the number of Android users ($\simeq$ 2B) and apps ($\simeq$ 2M). Thus, we generated a network with 500K users (125K per each user type) and 500 apps. We set the number of permissions per app and number of apps per user to be 5 and 2 respectively. The number of users per apps follows the *Long-tail* distribution. To generate such dataset of users/apps and assign the apps to the users, we used the *power-law* distribution formulation. Power-law distribution formula is $f(cx) = a(cx)^{-k} = c^{-k} f(x) \propto f(x)$, in which x denotes the range of the distribution, a denotes the normalization constant (maximum popularity), k denotes the distribution shape parameter and c is a constant value for scaling the distribution. Fig. 10(a) shows the distribution of users and apps. For the sake of clarity, only a part of the distribution is illustrated. In this configuration, the number of the most (app 1) and least (app 500) popular apps' users are $\approx$500K and $\approx$11 respectively.

Figs. 10(b) and Figs. 10(c) show the coverage and accuracy of DroidNet under different τ_e and τ_d settings. We can see that with lower values of τ_d (Algorithm 2), the coverage is higher while the accuracy is lower. This shows the trade-off between the coverage and accuracy. We also notice that the accuracy increases with increasing values of the experts filtering threshold τ_e. However, with very high τ_e, the coverage is low. This is because when all users are included in the decision process, the conflict of responses among users leads to low voting score and therefore DroidNet is less likely to make recommendations. In this experiment, the seed experts coverage is set to 1% of the apps.

To evaluate the relationship between the seed expert coverage and false positive on user classification, we repeated the last experiment under 5 different seed expert coverage rates, while using the same configuration. In this experiment, we also show the difference between two scenarios in terms of covering apps by seed experts. First, when seed experts cover apps randomly selected among top 10% apps and second, when seed experts cover the top 1% apps. It is worth mentioning that the τ_e is set to 0.9 in both scenarios. As shown in Fig. 10(d)(e), the number of users assigned to high rating group increases when the seed expert coverage increases in both scenarios. The main difference is that using the long-tail distribution helps classifying users more accurately (Fig. 10(e)).

In the last experiment, we also compare the performance of DroidNet and the PMP system [5]. Fig. 10(c) shows that the PMP achieves the accuracy of 0.57, whereas DroidNet's accuracy is higher than 0.8.

VII. Usability Evaluation

We also evaluated DroidNet through real user experiments. We collected user data to measure the accuracy, reliability, ease-of-use and practicality of the system described as follows.

A. Participants

We recruited 100 real users to participate in our experiments. To introduce diversity of the participants, we have

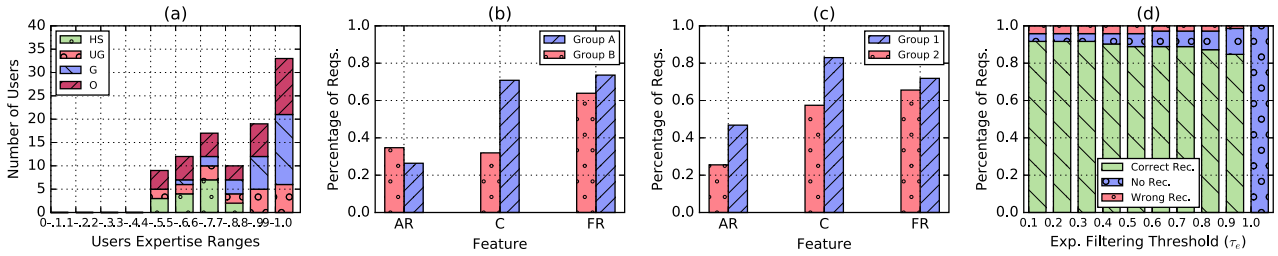


Fig. 11. Correctness and recommendation positive response/following rates: (a) expertise ratings of participants; (b) users responses analysis; (c) applications' permission requests analysis; (d) recommendations' accuracy under different normalized τ_e and a fixed $\tau_d = 0.5$.

TABLE II
DIVERSITY OF PARTICIPANTS (EDUCATION LEVEL)

Educational Level	High-School	Undergrad	Grad	Others
Number of users	16	20	28	36

TABLE III
DIVERSITY OF PARTICIPANTS (AGE)

Age range	18 – 19	19 – 25	25 – 35	35 – 50
Number of users	21	35	37	7

users from different educational levels (High-School (HS), Undergraduate (UG), Graduate (G) and Others (O)). Table II shows the diversity of the participants in our experiments with respect to educational levels. The participants with a higher education degree (undergraduate and graduate) were selected from different majors (engineering and non-engineering). We purposely selected 20 of the grad participants from Computer Science major to see whether participants' majors matters or not. Table III also shows the diversity of our participant in terms of age ranges. We hired participants from different ages (18-50) to conduct a comprehensive usability experiment. It is worth noting that 38% and 62% of our participants are female and male respectively.

B. Applications

We selected 12 applications to be evaluated in our experiments. The selected apps include 6 "*trusted*" apps (top ranked) and 6 *aggressive* (not ranked) apps. We define *trusted* apps to be those developed by trusted developers such as *Instagram* (social network), *Weather Channel* (weather category), etc. We define *aggressive* apps to be the apps that request irrelevant resources that they do not need. The apps request various resources including Internet communication, location, camera, storage (photos/media/files), SMS service, and user's contacts. We selected apps from different app categories such as communication, social network, finance, weather, music-audio, card game, and arcade. In each of these categories we downloaded a pair (trusted and aggressive) of apps. There are 72 permission requests in total.

C. Devices and OS

To prepare for the experiment, we built a customized Android ROM (Android 4.3, Jelly Bean) equipped with DroidNet system. We have used 4 LG Nexus 4 devices running DroidNet system.

D. Server

DroidNet records all responses from users and stores them on an online server. The server is implemented on a LAMP stack and uses CentOS, Apache, SQLite3, and the latest version of php.

We asked all participants to respond to the permission requests independently. Among the 72 requests, only 30 requests have DroidNet recommendations available. Those recommendations are created on purpose and may be incorrect to test how likely users will follow those recommendations blindly. We collected the responses from all users for analysis. The ground truth of all permission requests were provided by our seed expert.

1) Data Analysis: Before presenting the results, we show an overall view of the expertise level of users. Fig. 11(a) shows the expertise rating (unormalized) of all participants based on our collected data and ground truth provided by our seed expert. After applying our expertise rating algorithm on the recorded responses, the expertise rating results of users range from 0.1 (lowest) to 0.92 (highest). Considering the calculated users' expertise ratings, we classify them into four types, users with very low (< 0.1), low (0.1-0.5), medium (0.5-0.7), and high (> 0.7) level of expertise. In this figure, we can see that participants with higher level of education have higher expertise level. As we described in the participants demographic section, 20 of the participants had graduate degrees. 15 of the grad participants have expertise level between $0.9 - 1$. Out of this number 13 of them are Computer Science students. From this result, we can prove that educational background has a direct relation with the expertise level.

To study the correlation between user behavior and their expertise level, we selected two groups of users. Group A are users who are savvy (expertise rating higher than 0.8) and group B are inexperienced (expertise rating below 0.5). Fig. 11(b) shows the responses from users from the two groups. The *correctness rate (C)* is the percentage of correctly answered requests, and the *following rate (FR)* is the percentage of requests which followed the DroidNet recommendations. We see a strong correlation between the correctness rate and users groups and a weak correlation between the following rate and user groups. Users in group A achieves much higher correctness rate and behave more conservative

TABLE IV
USERS' OPINION ON DATA AND DEVICE SECURITY

Device Security	Secure	Neutral	Not secure	Total
Privacy Concern				
Concerned	28%	23%	15%	66%
Neutral	11%	7%	3%	21%
Not concerned	3%	7%	3%	13%
Total	42%	37%	21%	100%

TABLE V
DROIDNET'S TRUSTWORTHINESS AND EASE-OF-USE

Level	Low	Medium	High
Feature			
Ease-of-Use	2%	14%	84%
Trustworthiness	8%	20%	72%

in terms of granting permissions. In other words, the *accept rate (AR)* is lower.

We also divided apps into two different groups: trusted group (group 1) and aggressive group (group 2). Fig. 11(c) shows the responses received for both types of apps. We can see that requests from the apps in group 1 are more likely to be accepted by users, and a higher accuracy rate is also observed for trusted apps. There is no strong correlation between the following rate and app groups.

To evaluate the effectiveness of DroidNet on real data, we run the DroidNet recommendation algorithm on the collected real user data with parameter setting $\tau_d = 0.5$. Note that the recommendations are made only based on the users responses by ignoring the seed experts and normalizing expertise ratings. Fig. 11(d) shows that the percentage of incorrect recommendations decreases, while the cases of no recommendation increases for increasing values of τ_e. When τ_e is too high, no recommendation will be made.

2) Survey Statistics: Along with the real data collection, we also conducted a survey to measure different factors of DroidNet. In addition to participating in our test, we asked all the 100 users to fill in a questionnaire and answer to some objective multiple-choice questions. Table IV shows that the majority percentage (66%) people are concerned with their data privacy on mobile phones, while a large percentage (42%) people believe that the smart phone they use is secure.

We also surveyed the Ease-of-Use and trustworthiness of the DroidNet system. Table V shows that the majority (84%) of the participants believed that DroidNet is easy to use. 72% of the users think that DroidNet's recommendations are reliable.

VIII. THREATS AND DEFENSES

Although the purpose of DroidNet is to protect inexperienced smartphone users from being attacked by malicious apps, DroidNet itself may be the target of attacks. In this section, we discuss a few potential threats to DroidNet that we can foresee at this stage. We then show that through integrating strategic defensive design into DroidNet framework, we can detect, deter, or mitigate such threats. We also address the privacy concerns which may rise from DroidNet users and we show that our privacy-aware data collection design can reduce this concern to a minimum.

A. Injecting False Recommendations

One of the main important threats is the injection of false responses to mislead the recommendation system. For example, during the external expert users seeking process, malicious users/attackers behave well in order to be rated as expert users. After being chosen as expert users, they turn around and suggest dishonest recommendations to mislead the recommendation system.

We have investigated this potential threat and developed a multi-agent game theory model to study the gain and loss of malicious user and the DroidNet defense system. We derived a system configuration to discourage rational attackers to launch such attacks. Through the proposed game model we analyzed the interaction (request/response) between DroidNet users and DroidNet system using a static Bayesian game formulation. In the game, both the DroidNet system and attackers choose the best response strategy to maximize their expected payoff and we studied the Nash Equilibria of the game. We also identified the strategies that DroidNet can use to disincentivize attackers in the system, since they have no gain by attacking the system.

B. Bot Users

Bot users are fake users which are set up and controlled by attackers to fulfill some specific purpose. For example, the vendor of a malicious app may create many "expert" DroidNet users who will be honest when responding to other applications except to the particular app owned by their "master". Since DroidNet heavily relies on the responses from expert users, many dishonest expert users may misguide DroidNet into providing wrong recommendations if not detected and handled properly. How to detect those bot users and mitigate their impact is an important problem for DroidNet.

In order to address this issue, we have developed a clustering-based method for finding groups of bot users controlled by the same masters, which can be used to detect bot users with high reputation scores. The key part of the proposed method is to map the users into a graph based on their similarity (*features*) and apply a clustering algorithm to group users together. Specifically, we found that malicious users controlled by the same master may: (i) download and respond to the malicious app at as soon as the app is available; (ii) have unusual high overlap on the apps they installed and responded since they are from the same master; (iii) respond to the malicious app differently than benign expert users. We consider the above three features as behaviors of malicious (bot) users. We then apply a customized *weighted* distance function to aggregate them into the similarity between users.

C. Application Crashing and DroidNet's Overhead

In the current implementation of DroidNet, we created an OS patch from all the modifications. Since users have to apply the OS patch to be able to use DroidNet service, it may not

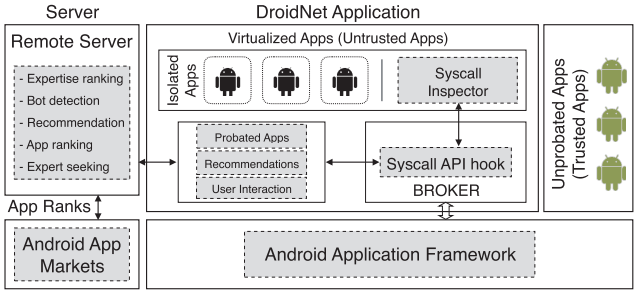


Fig. 12. DroidNet implementation architecture overview.

be practical for most users. Therefore, as a secondary implementation plan, our framework can be also implemented at the application level in order to make the service accessible to the majority of users by installing an app. This implementation can also avoid app crashing in case of permission denial.

As our future plan, we can utilize Android *isolated processes* to be able to run apps at the application level. Figure 12 shows the application-level implementation architecture. In this implementation isolated processes can be utilized to virtualize apps by loading them into the DroidNet and execute them. This way, system calls and permission requests can be captured by a component called Broker using Android internal/hidden APIs. The Broker is a system call API hook between the application level and the Android framework. In this implementation users only need to install the *DroidNet app* that can be deployed without firmware modifications or root privileges. One promising example of utilizing Android isolated processes can be Boxify [10]. Boxify is a solution that runs apps at the application level. Boxify has been evaluated and the evaluation results demonstrate its capability to enforce permission control without incurring a significant runtime performance overhead [10]. In addition, using this implementation, apps do not crash upon permission denial, which is an important improvement compared to the current implementation.

Intercepting API calls and syscalls, and enforcing permission control imposes some performance overhead. As one of the main use-cases, Boxify facilitates fine-grained permission control because of it low overhead and runtime robustness. As reported [10], the performance of Boxify is evaluated through a prototype containing both API calls and syscalls. Intercepting API calls to the application framework imposes ans overhead around 1%. For syscalls, a $\approx 100\mu s$ overhead is observed. However, the employed evaluation benchmark depicts the worst case scenario and the overall performance impact on apps is much lower. The measured overall performance overhead by executing several benchmarking apps on top of Boxify, is an acceptable degradation of $1.6\% - 4.8\%$. Therefore, since DroidNet's main activities are calls interception and permission enforcement, we can conclude that Boxify is an effective platform to implement DroidNet on top of it.

D. Privacy Concerns

DroidNet is a crowdsourcing-based solution and seeking expert users in the network is an important task.

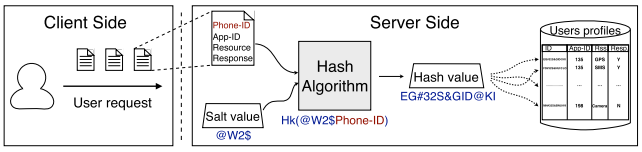


Fig. 13. Use one-way ID hashing to protect users' privacy.

DroidNet collects permission responses from all participating users to discover experts. To protect the privacy of users, we design a privacy-aware data collection mechanism that uses *hashing and salting* method (Figure 13) to protect the true identity of the users. The salt is randomly generated upon installation. Note that this mechanism provides *double-blind* protection, which means that an attacker who successfully attacked the database will not be able to reverse the function to find out the real phone ID or even verify whether an given phone ID is in the database. Therefore, the identity of the users is well-protected, and our mechanism does not compromise the usability of the collected data.

E. Platform Dependency

A well-designed model is a model that is independent of the specific technological platform used to implement it. Such model is called platform-independent. In other words, a platform independent model performs effectively and it is not restricted by the type of environment provided. We believe that DroidNet is a platform-independent model that can be applied and implemented on various mobile operating systems and hardware platforms. In the case of Android OS updates (at any layer), since the DroidNet model itself is platform-independent, so it can be adapted to the updates. In other words, as long as Android is using a permission-based mechanism as one of its security mechanism, DroidNet can be applied to it.

IX. Conclusion

In this paper we present DroidNet, a Android permission control and recommendation system which serves the goal of helping users perform low-risk resource accessing control on untrusted apps to protect their privacy and potentially improve efficiency of resource usages. We propose a framework that allows users to install apps in either trusted mode or probation mode. In the probation mode, users are prompted with resource accessing requests and make decisions on whether to grant the permissions or not. To assist inexperienced users to make low-risk decisions, DroidNet provides recommendations on permission granting based on the responses from expert users in the system. In order to do so, DroidNet uses crowdsourcing techniques to search for expert users using a transitive Bayesian inference model. Our evaluation results demonstrate that DroidNet can effectively locate expert users in the system through a small set of seed experts. The recommending algorithm achieves high accuracy and good coverage when parameters are carefully selected. We implemented our system on Android phones and demonstrate that the system is feasible and effective through real users experiments.

REFERENCES

[1] *What is the Price of Free*. Accessed: Oct. 1, 2017. [Online]. Available: http://www.cam.ac.uk/research/news/what-is-the-price-of-free

[2] *Apps by Downloads: Download Distribution of Android Apps*. Accessed: Aug. 2017. [Online]. Available: https://www.appbrain.com/stats/android-app-downloads

[3] *Download Statistics: Distribution of Downloads in the Android Market*. Accessed: Aug. 2017. [Online]. Available: http://www.androlib.com/appstatsdownloads.aspx

[4] *Gartner: 1.1 Billion Android Smartphones, Tablets Expected to Ship in 2014*. Accessed: May 2015. [Online]. Available: http://www.reuters.com/article/us-mobile-devices-gartner/more-than-1-billion-android-devicesto-ship-in-2014-gartner-idUSBREA060E220140107

[5] Y. Agarwal and M. Hall, "ProtectMyPrivacy: Detecting and mitigating privacy leaks on iOS devices using crowdsourcing," in *Proc. 11th Annu. Int. Conf. Mobile Syst., Appl., Services (MobiSys)*, New York, NY, USA, 2013, pp. 97–110.

[6] E. Aldhahri, V. Shandilya, and S. Shiva, "Towards an effective crowdsourcing recommendation system: A survey of the state-of-the-art," in *Proc. IEEE Symp. Service-Oriented Syst. Eng.*, Mar. 2015, pp. 372–377.

[7] R. Amadeo. *App Ops: Android 4.3's Hidden App Permission Manager, Control Permissions for Individual Apps!* [Online]. Available: http://www.androidpolice.com/2013/07/25/app-ops-android-4-3s-hidden-app-permission-manager-control-permissions-for-individual-apps/

[8] V. Ambati, S. Vogel, and J. Carbonell, "Towards task recommendation in micro-task markets," in *Proc. 11th AAAI Conf. Human Comput. (AAAIWS)*, 2011, pp. 80–83.

[9] S. Amini, "Analyzing mobile App privacy using computation and crowdsourcing," Ph.D. dissertation, Dept. Elect. Comput. Eng., Carnegie Mellon Univ., Pittsburgh, PA, USA, 2014.

[10] M. Backes, S. Bugiel, C. Hammer, O. Schranz, and P. von Styp-Rekowsky, "Boxify: Full-fledged app sandboxing for stock Android," in *Proc. 24th USENIX Secur. Symp. (USENIX Security)*, Washington, DC, USA, Aug. 2015, pp. 691–706.

[11] A. R. Beresford, A. Rice, N. Skehin, and R. Sohan, "MockDroid: Trading privacy for application functionality on smartphones," in *Proc. HotMobile*, 2011, pp. 49–54.

[12] D. C. Brabham, *Crowdsourcing*. Hoboken, NJ, USA: Wiley, 2013.

[13] G. W. Brown and J. W. Tukey, "Some distributions of sample means," *Ann. Math. Statist.*, vol. 17, no. 1, pp. 1–12, 1946.

[14] M. Dietz, S. Shekhar, Y. Pisetsky, A. Shu, and D. S. Wallach, "Quire: Lightweight provenance for smart phone operating systems," in *Proc. USENIX Secur. Symp.*, 2011, pp. 347–362.

[15] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android permissions demystified," in *Proc. 18th CCS*, 2011, pp. 627–638.

[16] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner, "Android permissions: User attention, comprehension, and behavior," in *Proc. SOUPS*, 2012, pp. 3:1–3:14.

[17] C.-S. Hwang, Y.-C. Su, and K.-C. Tseng, "Using genetic algorithms for personalized recommendation," in *Computational Collective Intelligence. Technologies and Applications*. Berlin, Germany: Springer, 2010, pp. 104–112.

[18] Q. Ismail, T. Ahmed, A. Kapadia, and M. K. Reiter, "Crowdsourced exploration of security configurations," in *Proc. 33rd Annu. ACM Conf. Human Factors Comput. Syst. (CHI)*, New York, NY, USA, 2015, pp. 467–476.

[19] M. Lange, S. Liebergeld, A. Lackorzynski, A. Warg, and M. Peter, "L4Android: A generic operating system framework for secure smartphones," in *Proc. SPSM*, New York, NY, USA, 2011, pp. 39–50.

[20] J. Lin, B. Liu, N. Sadeh, and J. I. Hong, "Modeling users' mobile app privacy preferences: Restoring usability in a sea of permission settings," in *Proc. Symp. Usable Privacy Security (SOUPS)*, Menlo Park, CA, USA, Jul. 2014, pp. 199–212.

[21] B. Liu, J. Lin, and N. Sadeh, "Reconciling mobile app privacy and usability on smartphones: Could user privacy profiles help?" in *Proc. 23rd Int. Conf. World Wide Web (WWW)*, New York, NY, USA, 2014, pp. 201–212.

[22] B. Liu, S. Nath, R. Govindan, and J. Liu, "DECAF: Detecting and characterizing ad fraud in mobile apps," in *Proc. 11th USENIX NSDI*, Berkeley, CA, USA, 2014, pp. 57–70.

[23] R. Liu, J. Cao, L. Yang, and K. Zhang, "PriWe: Recommendation for privacy settings of mobile apps based on crowdsourced users' expectations," in *Proc. IEEE Int. Conf. Mobile Services*, Jun. 2015, pp. 150–157.

[24] K. Meehan, T. Lunney, K. Curran, and A. McCaughey, "Context-aware intelligent recommendation system for tourism," in *Proc. IEEE Int. Conf. Pervas. Comput. Commun. Workshops (PERCOM Workshops)*, Mar. 2013, pp. 328–331.

[25] R. Mittal, A. Kansal, and R. Chandra, "Empowering developers to estimate app energy consumption," in *Proc. 18th Mobicom*, New York, NY, USA, 2012, pp. 317–328.

[26] O. R. E. Pereira and J. J. P. C. Rodrigues, "Survey and analysis of current mobile learning applications and technologies," *ACM Comput. Surveys*, vol. 46, no. 2, pp. 27:1–27:35, Dec. 2013.

[27] B. Rashidi, C. Fung, A. Nguyen, and T. Vu, "Android permission recommendation using transitive Bayesian inference model," in *Proc. 21st Eur. Symp. Res. Comput. Secur. (ESORICS)*, 2016, pp. 477–497.

[28] F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor, Eds. New York, NY, USA: Springer, 2011, pp. 1–35.

[29] W. Rothman. *Smart Phone Malware: The Six Worst Offenders*. Accessed: Oct. 1, 2017. [Online]. Available: http://www.nbcnews.com/tech/mobile/smart-phone-malware-six-worst-offenders-f125248

[30] Statista. *Number of Apps Available in Leading App Stores as of June 2016*. Accessed: Oct. 1, 2017. [Online]. Available: http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/

[31] X. Wang, L. Mudie, and C. J. Brady, "Crowdsourcing: An overview and applications to ophthalmology," *Current Opinion Ophthalmol.*, vol. 27, no. 3, pp. 256–261, 2016.

[32] L. Yang, N. Boushehrinejadmoradi, P. Roy, V. Ganapathy, and L. Iftode, "Short paper: Enhancing users' comprehension of Android permissions," in *Proc. 2nd ACM Workshop Secur. Privacy Smartphones Mobile Devices (SPSM)*, New York, NY, USA, 2012, pp. 21–26.

[33] M.-C. Yuen, I. King, and K.-S. Leung, "Task matching in crowdsourcing," in *Proc. Int. Conf. Internet Things 4th Int. Conf. Cyber, Phys. Soc. Comput.*, Oct. 2011, pp. 409–412.

Bahman Rashidi (S'16) received the master's degree in computer engineering from the Iran University of Science and Technology, Tehran, Iran, in 2014. He is currently pursuing the Ph.D. degree in computer science with the Virginia Commonwealth University. His research interests include distributed systems, mobile systems, and privacy issues in smartphone devices. He was a recipient of the Outstanding Early-Career Student Researcher Award from the VCU Computer Science Department in 2015.

Carol Fung received the bachelor's and master's degrees in computer science from the University of Manitoba, Canada, and the Ph.D. degree in computer science from the University of Waterloo, Canada. Her research interests include collaborative intrusion detection networks, social networks, security issues in mobile networks and medical systems, security issues in next generation networking, and machine learning in intrusion detection.

Anh Nguyen is currently pursuing the Ph.D. degree with the Computer Science and Engineering Department, University of Colorado Denver. She has been a Research Assistant with the Mobile and Networked Systems Lab, since 2014. Her research in mobile computing systems focuses on mobile healthcare, mobile sensing, hardware development, and mobile applications. She is also interested in signal processing and pattern recognition.

Tam Vu received the B.S degree in computer science from the Hanoi University of Technology, Vietnam, in 2006, and the Ph.D. degree in CS from the WINLAB, CS Department, Rutgers University, in 2013. He is currently an Assistant Professor and the Director of the Mobile and Networked Systems Lab, the CS Department, University of Colorado Denver. He received the Google Faculty Research Award in 2014. He received the Best Paper Award from the ACM MobiCom 2011 and the ACM MobiCom 2012.

Elisa Bertino (F'02) was a Professor and the Department Head with the Department of Computer Science and Communication, University of Milan. She was a Visiting Researcher with the IBM Research Laboratory, Microelectronics and Computer Technology Corporation, Rutgers University, Telcordia Technologies. She is a Professor of computer science with Purdue University. She is a fellow of the ACM. She received the IEEE Computer Society 2002 Technical Achievement Award and the IEEE Computer Society 2005 Kanai Award. She serves as Editor-in-Chief for the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING.