Android Permission Recommendation Using Transitive Bayesian Inference Model

Bahman Rashidi[1](✉), Carol Fung[1], Anh Nguyen[2], and Tam Vu[2]

[1] Virginia Commonwealth University, Richmond, VA 23284, USA
{rashidib,cfung}@vcu.edu

[2] University of Colorado Denver, Denver, CO 80202, USA
{anh.t4.nguyen,tam.vu}@ucdenver.edu

Abstract. In current Android architecture, users have to decide whether an app is safe to use or not. Technical-savvy users can make correct decisions to avoid unnecessary privacy breach. However, most users may have difficulty to make correct decisions. DroidNet is an Android permission recommendation framework based on crowdsourcing. In this framework, DroidNet runs new apps under probation mode without granting their permission requests up-front. 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 rating algorithm using transitional Bayesian inference model. The recommendation is based on the aggregated expert responses and its confidence level. Our evaluation results demonstrate that given sufficient number of experts in the network, DroidNet can provide accurate recommendations and cover majority of app requests given a small coverage from a small set of initial experts.

1 Introduction

As the population of smartphone users continues to grow, smartphones have brought significant impact to businesses, social, and lifestyle. With the expectation of over 10 billion mobile Internet devices by 2016, the mobile application industry is putting forward tremendous effort to match the demand and keep up with the ever-evolving technologies [17]. On the other hand, the number of mobile apps has been growing exponentially in the past few years. According to the report by Android Google Play Store, the number of apps in the store has reached 1.8 billion in 2015, surpassing its major competitor Apple App Store [22].

Unlike iOS, Android device owners do not have to root or "jailbreak" their devices to install apps from "unknown sources". This gives Android users broad capability to install pirated, corrupted or banned apps from Google Play simply by changing a systems setting. This provides further incentive for the users to install third-party applications from various (potentially untrusted) app markets [2], but exposes their privacy to significant security risks [3].

In the current Android architecture (6.0), users decide what resources are given to an app by responding to permission requests from apps. Users can also

I. Askoxylakis et al. (Eds.): ESORICS 2016, Part I, LNCS 9878, pp. 477–497, 2016.
DOI: 10.1007/978-3-319-45744-4_24

manually manage any app's permissions after installation. However, this permission control mechanism has been proven to be ineffective to protect users from malicious apps. Study shows that more than 70 % of smartphone apps request to collect data irrelevant to the main functionality of the app [1]. In addition, study shows that only a very small portion (3 %) of users pay attention and make correct decisions to the resource being requested at installation time, since they tend to rush through to get to use the application. The current Android permission pop-ups (Android 6) still depends on users for security and privacy decisions, which is not reliable since inexperienced users may not be able to or care to make correct decisions.

As pointed out in [11,12], 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 obliged to exchange their privacy for using the app. To address these problems, we propose DroidNet, a framework to assist mobile users to control their resource usage and privacy through crowdsourcing-based permission control recommendations. First, the framework allows users to use apps without granting all permissions. Second, DroidNet allows receiving 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.

The key challenge is to expand the expert user base so that their expertise can cover all applications on the market. More importantly, these expert users should be selected so that their responses and recommendation to their peer of high quality. DroidNet starts from a small set of trusted expert users and propagates the expert evaluation using a transitional Bayesian learning model. We evaluate the effectiveness of the model through a set of experiments. 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 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 evaluation on the reliability of the system.

The rest of the paper is organized as following. We first discuss the existing literature in resource management and permission controls in the next section. We then discuss the DroidNet's system view in Sect. 3, followed by the presentation of our algorithms in Sect. 4. Sections 5 and 6 present our implementation and evaluation results respectively. Section 6 shows our evaluation results brought about by our real experiments. We conclude this paper by a discussion and conclusion of our work.

2 Related Work

Due to its inherent constraints in resources, much effort has been done towards the principles and practices to manage resource usage and privacy protection [8,13] of mobile applications. However, the most common practice for resource access management today is Mandatory Access Control (MAC) mechanism [10,21], which is found in API from major mobile players such as iPhone, Android, and Windows Phone. In such paradigm, resource access from apps needs to be granted by users. In Android, this is done through its *Static Permission Model* [14] where users need to grant all requested permissions on installation.

Kathy Wain Yee et al. proposed PScout and studied the design implications of this model and performed an analysis to extract the permission specification from the Android OS [6]. The large number of permissions and APIs in Android suggests that the permission specification for Android is complicated. They also found out that documented and undocumented APIs are heavily interconnected.

Studies [11,12] have shown that such permission control paradigms are not efficient since users are either not paying attention to permissions being requested or not aware of the permissions' implications.

Revising the current Android framework and/or runtime to provide *fine-grained permission controls*, AppFence [15], MockDroid [9], and TISSA [25] avoid giving out sensitive data by granting fake permissions. While such approaches reduce the risk of leaking private information and critical resources, it requires users to make decisions on *every resource request*, which is difficult for inexperienced users and time consuming. Alternatively, AppGuards [7], Aurasium [23], and FireDroid [20] allow users to define security policies on Android apps in which the policies are enforced through their framework. In FireDroid, an application monitor is created to track all processes spawned in Android and allow/deny them based on human managed policies. This approach requires *rooting* the device and extracting the Android booting partition, which is not practical for most users. In addition, the policy file editing and management are also difficult for most inexperienced users. Therefore, FireDroid targets on corporation phone users where the FireDroid can be pre-installed and policies are managed by administrators. Similarly, AppGuards and Aurasium require users to define security policies which is not practical for inexperienced users. In contrast DroidNet is designed for inexperienced users since it does not require users to have prior knowledge or be technical savvy to use it.

Exploring user perceptions of privacy on smartphones using crowdsourcing has been studied in the literature. Agarwal et al. propose PMP [4] which collects users' privacy protection decisions and analyses them to recommend them to other iOS users. However, their recommendations are based on simple *majority opinion* condition which causes high false recommendation rate. In contrast, we propose an expertise ranking algorithm to evaluate the expertise level of users for a higher quality recommendation. Ismail et al. propose a crowdsourcing solution to find a minimal set of permissions that will preserve the usability of the app for diverse users [16]. Their proposed work has a few shortcomings. Repackaging apps for all possible permission combinations is not plausible in reality. Also their indifference among inexperienced and malicious users makes their recommendations with limited quality. Yang et al. [24] 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. Android 4.3 (App Ops [5]) and 6's permission control mechanisms allow users to selectively disable permissions for apps on their phones.

In our previous work RecDroid [18,19], we proposed to utilize experts users' decisions to help inexperienced users on Android permission control using a simple expert rating and recommendation algorithm. However, only users who have direct overlap with seed experts will be rated, which largely limits its coverage in a large network. In this paper, we developed a network-based expert seeking and a decision recommendation algorithm based on *transitive Bayesian inference theory*, with which more users can be rated through the transition model. Furthermore, we implemented the system on Android platform and conducted a set of comprehensive experiments to evaluate the performance and reliability of the system.

3 System Design

DroidNet has 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) analyses the responses to eliminate untruthful and biased responses, (3) suggests other users with low-risk responses to permission requests, and (4) ranks apps based on their security and privacy risk level inferred from users' responses. Figure 1 shows an overview of DroidNet architecture, which is composed of a *thin OS patch* allowing mobile clients to automatically report users' responses to and receive permission request response suggestions from a *DroidNet* service. The differentiating factor of DroidNet is the ability to seek expert user base on a small set of seed users (Sect. 4). In the rest of this section, we describe four key features of the DroidNet system.

Permission handling. When installing an app, package installer needs to request permission to access resources on the device. Instead of sending requests to the Android system's legacy permission handler (e.g. Package Manager Service), DroidNet handles the permission requests through the logistics illustrated in Fig. 2. At the installation process of apps, DroidNet allows users to install the apps on one of the two modes: *Probation* mode and *Trusted* mode. On *Probation* mode, DroidNet closely monitors the requests to access a list of user-defined critical permissions, such as location access, contact list access, and camera access, during the application execution. When those resources are requested, a dialog box will pop up to guide users to make customized decision on whether the access to those critical resources should be granted to the app or not. Otherwise, on *Trusted* mode, all requested permissions are permanently granted to the app.

In order to capture the realtime interaction between the system and users, we designed a *Permission Control Portal* on the mobile devices to intercept

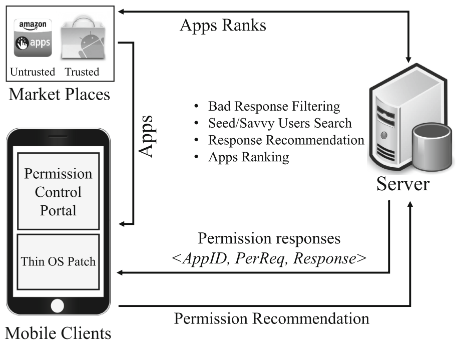

Fig. 1. DroidNet service overview

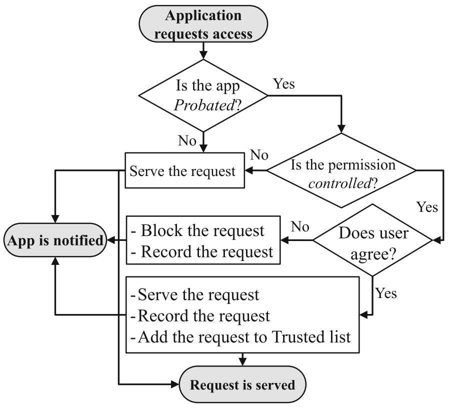

Fig. 2. Permission request flow in DroidNet

apps' permission requests, record the requests, and collect users' response to the requests. Since intercepting permission requests requires OS level access, we created a small software patch to modify client's operating system. We investigated different potential approaches to perform OS modification and designed a solution that causes minimum impact to legacy apps and applicable to a broad range of OS versions, hardware platforms, and permission access models.

Permission recommendation. To help inexperienced users with their decisions, DroidNet also provides recommended response to the users (Fig. 5 (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 responded with a random location. The user decisions are recorded by the DroidNet client and sent to the DroidNet server for further analysis. After that, the requests are forwarded to legacy permission handler for book keeping and minimizing DroidNet's unexpected impact on legacy apps. 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, 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 the usability of the system.

4 Expert Users Seeking

In DroidNet users' responses to permission requests are recorded by a central server and the responses from expert users are used to generate recommendation to help inexperienced users make low-risk decisions. However, an effective

Table 1. Notations

Notation	Description
$\mathcal{U}$	$\{U_1, ..., U_n\}$: Set of n DroidNet users in the system
s	The seed user set
$\mathcal{R}_i$	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 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

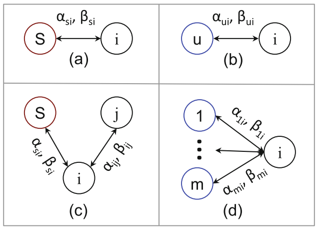

Fig. 3. The illustration of four cases of DroidNet graphs. (a) a user is connected directly to a seed user; (b) a user is connected to a non-seed user; (c) a multi-hop rating propagation case; (d) a multi-path rating aggregation case.

method to find expert users in the network is the key. DroidNet starts from a small set of trusted seed expert users, and propagate the expertise evaluation based on similarity among users using a transitive Bayesian inference model. This section describes the model in more detail.

4.1 Assumptions and Notations

The 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 $\mathcal{U} = \{U_1, U_2, ..., U_n\}$, and a set of edges $\mathcal{E} = \{e_{ij} | \mathcal{R}_i \cap \mathcal{R}_j \neq \emptyset, \forall i, j\}$ denoting users i and j have installed apps in common. Where $\mathcal{R}_i$ denotes the set of permission requests responded by user i.

The *seed expert* (SE) is one or a set of trusted expert users who might be employed by a DroidNet facilitator to provide accurate responses to permission requests. We assume seed experts are fully trusted and always provide correct response to permission requests of the apps they cover. 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 recommendation.

It is noteworthy that to make seeds' responses context-independent, they follow the principle of least privilege, which is finding a minimal set of permissions that are necessary for apps' legitimate purposes. This way, the seeds' responses are not depend on preference or context.

Let $\mathcal{R}_s$ denote the set of requests covered by seed experts. Then the common set of requests responded by both the seed user and user i can be written as $\mathcal{R}_{si} = \mathcal{R}_s \cap \mathcal{R}_i$. In general, the common set of requests responded by any two users i, j can be written as $\mathcal{R}_{ij} = \mathcal{R}_i \cap \mathcal{R}_j$. Table 1 lists the notations we use in this paper.

4.2 The Users Expertise Rating Problem

The key challenge of DroidNet is to seek experts from the regular users in the DroidNet so that the system can make low-risk recommendations based on the responses from those expert users. 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 the likelihood that a user i makes correct decision (p_i) satisfies a distribution Y_i with pdf $f_i(x)$. Then we define the expected 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 and a set of users $\mathcal{U} = \{U_1, \ldots, U_n\}$, a DroidNet graph is denoted as $\mathcal{G} = \{\mathcal{U} \cup s, \mathcal{E}\}$. Where $\mathcal{E} = \{e_{ij} | \mathcal{R}_i \cap \mathcal{R}_j \neq \emptyset, \forall i, j\}$ is a set of edges between users where they have overlap on responded requests. The expertise rating problem is to find the posterior distributions of all p_i, given their past history of responding to permission requests.

To present the solution of Problem 2, we first define the concept of similarity between a pair of users and then the computation method on expertise levels.

Definition 3. (*Similarity of Two Users*) Suppose user i and user j have responded to a common set of permission requests $\mathcal{R}_{ij}$, then we define the similarity between these two users using a *similarity tuple* (α_{ij}, β_{ij}), denoting 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 the two users provided consistent responses at the kth overlapped request, and vice versa. The similarity tuple can be computed as follows:

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

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

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 emphasize the importance of more recent observations.

4.3 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. 3(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 1. *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 Eqs. (3) and (4). □

4.4 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. 3(b)). To rate users who are connected only to a regular user with known expert rating, we can use the following theorem:

Theorem 4. *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. 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:

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

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

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 Eqs. (11) and (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 (5) and (6). □

4.5 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. 3(c), user i has overlap with the seed user, so it can be ranked through our Bayesian ranking algorithm described in Lemma 1. User j only has overlap with user i, so it can be ranked based on its similarity to user i. However, Theorem 4 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 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:*

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

Proof. From Eqs. (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:

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

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:

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

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:

$$\begin{aligned}
\alpha_i &= \alpha_i^1 + ... + \alpha_i^m = \sum_{k=1}^{m} \alpha_i^k \\
\beta_i &= \beta_i^1 + ... + \beta_i^m = \sum_{k=1}^{m} \beta_i^k \qquad (14)
\end{aligned}$$

$\square$

4.6 Multi-path User Rating Aggregation

A user may have overlap with multiple other users. As shown in Fig. 3(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 2. *Let i be a user who has overlap with a set of users $\mathcal{M} = \{U_1, U_2, ..., U_m\}$ with corresponding similarity tuples $\mathcal{S} = \{(\alpha_{1i}, \beta_{1i}), ..., (\alpha_{mi}, \beta_{mi})\}$. Then we have:*

$$\alpha_i = \alpha_i^1 + ... + \alpha_i^m = \sum_{k=1}^{m} \alpha_i^k$$
$$\beta_i = \beta_i^1 + ... + \beta_i^m = \sum_{k=1}^{m} \beta_i^k \tag{15}$$

where,

$$\alpha_i^k = \alpha_{ki} \frac{\alpha_k \alpha_i}{\alpha_k \alpha_i + \beta_k \beta_i} + \beta_{ki} \frac{\beta_k \alpha_i}{\beta_k \alpha_i + \alpha_k \beta_i}, \forall k \in \{1, 2, ..., m\}$$
$$\beta_i^k = \alpha_{ki} \frac{\beta_k \beta_i}{\alpha_k \alpha_i + \beta_k \beta_i} + \beta_{ki} \frac{\alpha_k \beta_i}{\beta_k \alpha_i + \alpha_k \beta_i}, \forall i \in \{1, 2, ..., m\}$$

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

Our approach to define the order of user rating is to start from the direct neighbors of the seed expert, 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.

4.7 Recommendation Algorithm

After rating users in the network, the next step 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 steps: qualification, voting, and decision. The algorithm is described in Algorithm 2.

In the qualification step, 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 ranking score of the voter. After the voting process finishes, the average ballot score is used to make a final decision. If the average ballot score exceeds a decision threshold, then corresponding recommendations are made. Otherwise, no recommendation is made.

5 Implementation

To prove the concept of feasibility, we implemented a prototype of DroidNet. We modified the permission management component of the Android operating system. We also provide users with an Android application to monitor and

Algorithm 1. Rate All Regular Users

```
1: Compute expertise rating of all regular users in DroidNet
2: Notations:
3: $R(\mathcal{U})$: the current rating of all users
4: $\hat{R}(\mathcal{U})$: the last round rating of all users
5: $s$: the seed expert
6: $U_i$: the $i_{th}$ user
7: $\mathcal{G} = (V, E)$: the generated graph of users and overlaps
8: $RU$: the set of rated users
9: $QU$: the queue of users to be rated
10: //parameters initialization
11: set $R(s) = 1$ and $R(U) = 0.5$
12: while $(Distance(R(\mathcal{U}), \hat{R}(\mathcal{U})) > \epsilon)$ do
13:     $RU \leftarrow s$
14:     $QU \leftarrow findNeighbors(s)$
15:     $\hat{R}(\mathcal{U}) \leftarrow R(\mathcal{U})$
16:     while $(u \leftarrow remove(QU)$ is not null) do
17:         //Users rating using Corollary 5.2
18:         $R(u) \leftarrow computeRating(u)$
19:         $RU \leftarrow RU \cup u$
20:         $\mathcal{N} = findNeighborsNotInRUorQU(u, \mathcal{G})$
21:         $push(\mathcal{N}, QU)$
22:     end while
23: end while
```

Algorithm 2. Vote for Recommendation

```
1: Notations :
2: $R(u), C(u)$ :the rating score and confidence of user $u$
3: $x(u)$ :the response to permission request from user $u$
4: $\tau_e, \tau_c$ :the minimum rating score and rating confidence to be considered as an expert user
5: $\tau_d$ :the recommendation threshold
6: $a, b$ :the cumulative ballots for yes or no decision
7: $D_0$ :the initial ballot count for both decisions
8: $a = b = D_0$
9: //Users filtering and ballots casting
10: for each user $u$ who responded to the request do
11:     if $R(u) > \tau_e$ and $C(u) > \tau_c$ then
12:         if $x(u) = 1$ then
13:             $a+ = R(u)$
14:         else
15:             $b+ = R(u)$
16:         end if
17:     end if
18: end for
19: //decision making based on final ballots count
20: if $\frac{a}{a+b} > \tau_d$ then
21:     Recommend to accept the request with confidence $\frac{a}{a+b} - \tau_d$
22: else if $\frac{a}{a+b} < 1 - \tau_d$ then
23:     Recommend to reject the request with confidence $1 - \frac{a}{a+b} - \tau_d$
24: else
25:     No recommendation
26: end if
```

manage resource access permissions at fine-grain level. Figure 4 illustrates DroidNet's implementation architecture. DroidNet is installed by applying a software patch which includes some modification on the Android operating system and a pre-installed app `DroidNet.apk` on the application level.

5.1 Permission Control User Interaction

The users of DroidNet have an option to install apps under a *probation mode*. We use the app "Telegram" (a popular messaging application) as an example. The first screenshot (Fig. 5(a)) displays two options when installing the app on the smartphone. They can be either *probation mode* or *trusted mode*.

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. 5(b)) where checked resources are monitored. By default all sensitive resources are monitored, and can be changed by users. 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. 5(c)).

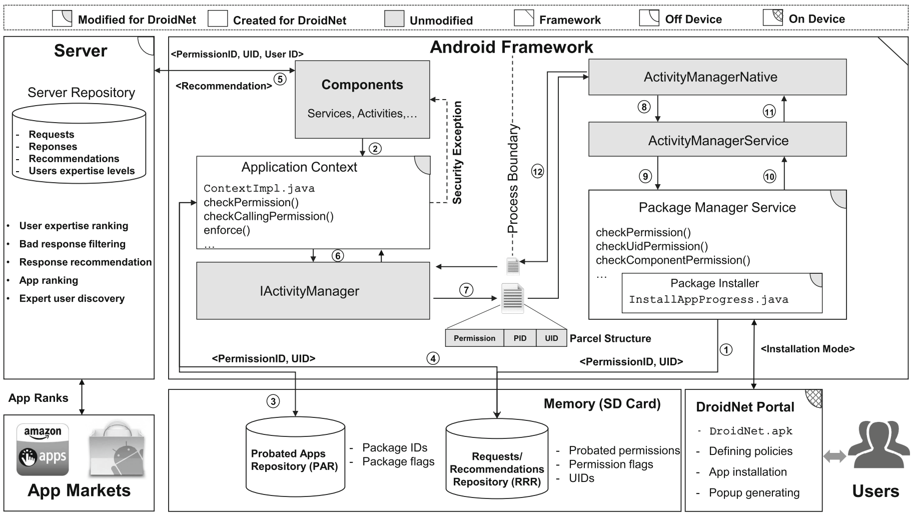

Fig. 4. DroidNet implementation architecture overview

5.2 Android OS 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 OS version 4.3 to meet our goal.

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 trusted mode installation then app would not be managed by DroidNet, and no information will be recorded about the application. If the user selects probation mode, DroidNet records app's UID and the set of requested permissions by probated app within the *Probated Apps Repository* (PAR) and *Request/Recommendation Repository* (RRR) repositories. Note that communication is obtained through using these repositories that all layers (framework and application) read and write from.

System Calls monitoring and Permission Enforcement: Our implementation is designed to be extensible and generic. While our implementation requires multiple changes in one place, it doesn't require modifications on every permission request handler, as it was the case on some previous works, such as in MockDroid [9]. The modification is presented in the form of an OS patch, which can be executed from a user's space, making this technique easier to adopt.

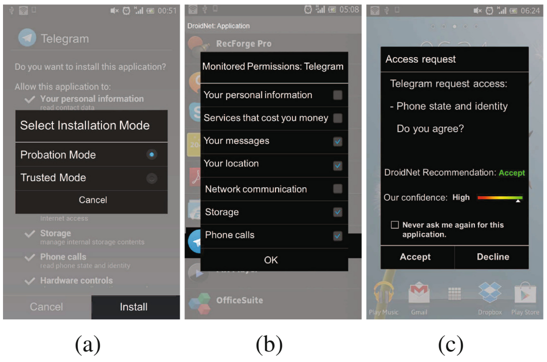

Fig. 5. An example of Line app: (a) probation and trusted installation modes; (b) users pick which critical resources to be monitored; (c) pop-up for permission granting with suggestion from DroidNet and its confidence.

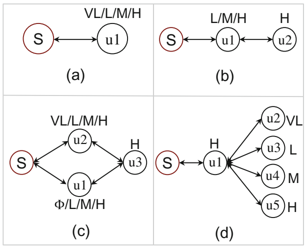

Fig. 6. The four user profiles designed for evaluation: (a) a user is connected directly to a seed user; (b) a user is away from seed by 1 hop; (c) a multi-path rating propagation case; (d) a multi-path rating case, designed for α and β calculation convergence.

In order to design an extensible and central permission enforcing point, we modified the `ContextImpl.java` class of the *context* component of the Android. This is called whenever an application seeks to use some permissions that are not hardware related. When this method is 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.

5.3 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.

6 Performance Evaluation

In this section we use simulation to evaluate the performance of the expert rating and recommendation algorithms. We also conduct a set of experiments to evaluate the reliability of the system.

6.1 Simulation Setup

As a proof of concept, we created a set of DroidNet user profiles consisting of four different types of expertise levels. 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. Our simulation environment is Visual Studio C++ on a Windows machine with 3.6 Ghz Intel Core i7 and 16 GB RAM. All results are based on an average of 500 repeated runs with different random generator seeds.

6.2 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 designed configuration. The average number of permission requests per app is set to 5 and the maximum number of requests is 500. Figure 6 shows the four study cases and their configurations.

We start from a simple case study that a user is connected to a seed expert only (Fig. 6(a)), and study the expertise ratings and rating confidence when the user is initialized with H, M, L and VL expertise ratings respectively. Figure 7(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. Figure 7(b) shows the corresponding confidence levels of estimation. The confidence 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 overlap with the seed expert.

In the second study case (Fig. 6(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 expertise. Figure 7(c) 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 study case (Fig. 6(c)), user 3 connects to the seed expert with two intermediate users. We vary the type of user 1 to be $\emptyset$ (non-exist), H, M, and L and vary the type of user 2 to be H, M, L, and VL. Figure 7(d) shows the *conductivity rule* of rating: high expert rating is propagable through expert intermediate nodes (conductive nodes).

Figure 7(e) shows the expertise rating of five users (with expertise $0.1, 0.5, 0.7,$ and 0.9) for study case shown in Fig. 6(d). As we described in Algorithm 1, we con-

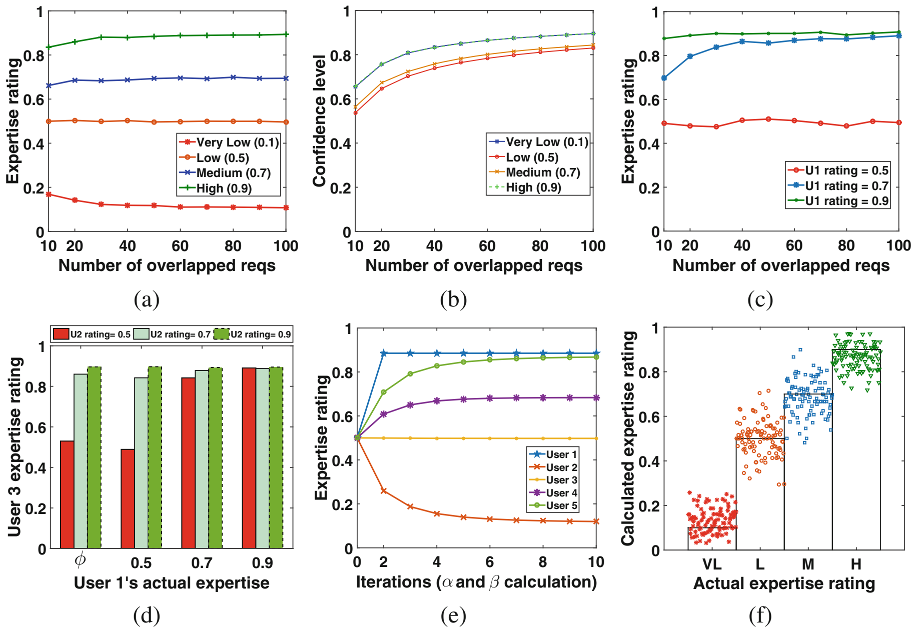

Fig. 7. User expertise and confidence level: (a) expertise level of user with different initial expertise level; (b) computed confidence level of user with different initial expertise level; (c) expertise rating of a user with only one user in its locality and different expertise ratings; (d) expertise rating of a user with two users in its locality and different expertise levels; (e) expertise rating of users for different number of α and β calculation iterations; (f) expertise rating distribution after rating users with different actual expertise rating.

tinue 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 this figure we can see that after 10 iterations of computing the 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 on a medium size network with 400 users and 250 apps in total. We set up 100 user for each type (VL, L, M and H). Users choose to install 20 apps out of 250 apps randomly. Figure 7(f) shows the distribution of the final expertise ratings for all 400 users. We can see that the estimated expertise ratings are clustered around their actual expertise levels and false positive and false negative exists.

To find the relationship between the seed expert coverage and false positive on user classification, we repeated the last experiment under 10 different coverage rate from seed experts, while remaining the same configuration otherwise. As shown in Fig. 8(a), the number of users assigned to high rating group increases when the seed expert coverage increases.

6.3 Quality of DroidNet Recommendation

We evaluate the quality of DroidNet recommendations using two metrics: *coverage* and *accuracy*. Coverage is the percentage of the requests that DroidNet can offer low-risk 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 recommend the response from the seed expert.

Our network setting consists of 250 apps with 5 requests per app (total 1250 requests). 400 users with four different types of users (100 users each type), and each user installed and responded to 20 apps (100 requests). Among the 250 apps, 20 of them are covered by the seed expert.

Figure 8(b) shows the coverage and accuracy of DroidNet under different τ_e and τ_d settings. We can see that with higher τ_d (Algorithm 2), the coverage increases while the accuracy decreases. This shows the trade-off between the coverage and accuracy. We also notice that the accuracy increases with experts filtering threshold τ_e. However, with very low or very high τ_e, the coverage is low. This is because when all users are included in the decision process, the conflict

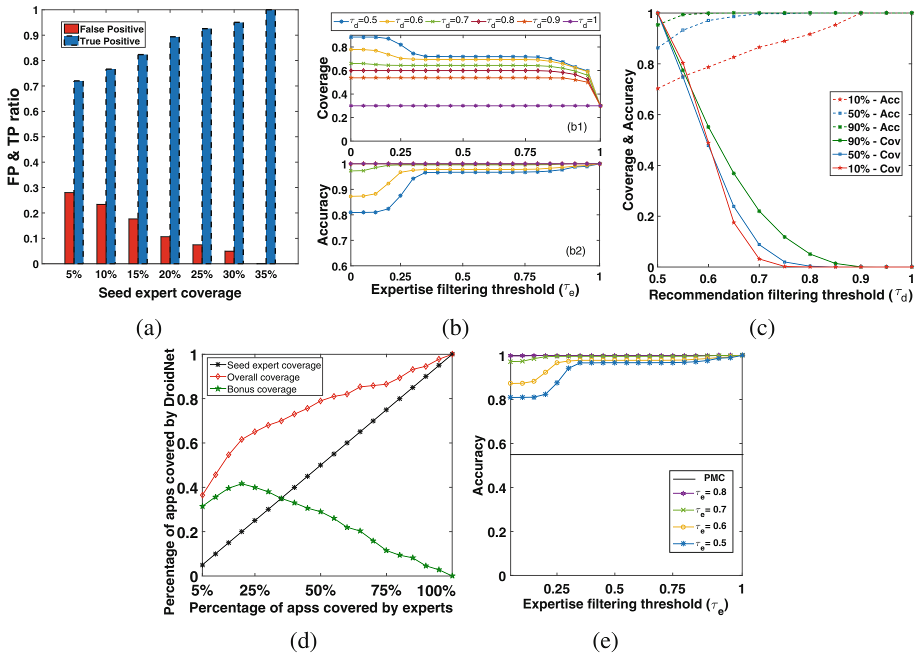

Fig. 8. Coverage and accuracy of rating and recommendation: (a) accuracy of rated users (high rating) and seed expert coverage relation; (b) percentage of requests that DroidNet makes recommendation; (c) percentage correct recommendations that DroidNet makes; (d) coverage of overall requests vs. coverage of seed experts; (e) PMP accuracy comparison.

of responses among users leads to low voting score and therefore DroidNet is less likely to make recommendations.

To study the impact from the number of expert users ($\tau_e = 0.9$) to DroidNet performance, we vary the percentage of expert users in the network (10 %, 50 % and 90 %). Figure 8(c) shows the results for the coverage and accuracy. We can see that DroidNet achieves higher coverage and accuracy with more expert users in the system.

Next we study the impact from seed expert coverage. As shown in Fig. 8(d), the overall DroidNet recommendation rate increases with coverage rate from the seed expert. The linear line represents the coverage rate from the seed user. The difference between the overall coverage and seed expert coverage is called the *bonus coverage*. Higher bonus coverage represents a higher utilization of DroidNet. From the economic point of view, if the coverage of seed expert brings cost to the coordinator (since the seed expert is hired), then the bonus coverage brings down cost to the coordinator. The decision makers can choose the optimal seed coverage based on its optimal profit.

In the last experiment, we compare the performance of DroidNet and the PMP system [4]. Figure 8(e) shows that the PMP achieves the accuracy of 0.47, whereas DroidNet's accuracy is higher than 0.8.

7 Discussion and Conclusion

While the major challenges for DroidNet have been discussed, many other ones still remain. In this section we address some potential issues in DroidNet and our solutions to those issues.

False Responses: One of the main threats to the system is the injection of false responses to mislead the recommendation system. We 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. Furthermore, we also consider enhancing DroidNet in which recommendations are adapted according to each user's different security and privacy needs.

Users' Privacy: As a crowdsourcing based solution, DroidNet collects permission responses from all participating users. To protect the privacy of users, we have designed a privacy-aware data collection mechanism that uses *hashing and salting* method to protect the true identity of the users. The salt is randomly generated upon installation. The solution provides *double-blind* protection, which means 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 are well-protected, and the mechanism does not compromise the usability of the collected data.

In summary DroidNet is an 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. The framework allows users to install apps in probation mode where 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 can achieve 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 user experiments.

References

1. What is the price of free. http://www.cam.ac.uk/research/news/what-is-the-price-of-free
2. F-droid - free and open source android app repository. https://f-droid.org/. Accessed August 2015
3. Bit9 report: pausing google play: more than 100,000 android apps may pose security risks. https://www.bit9.com/files/1/Pausing-Google-Play-October2012.pdf. Accessed May 2015
4. Agarwal, Y., Hall, M.: Protectmyprivacy: detecting and mitigating privacy leaks on IOS devices using crowdsourcing. In: Proceeding of the 11th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys 2013), New York, NY, USA, pp. 97–110. ACM (2013)
5. Amadeo, R.: App Ops: Android 4.3's hidden app permission manager, control permissions for individual apps! http://www.androidpolice.com/2013/07/25/app-ops-android-4-3s-hidden-app-permission-manager-control-permissions-for-individual-apps/
6. Au, K.W.Y., Zhou, Y.F., Huang, Z., Lie, D. Pscout: analyzing the android permission specification. In: Proceedings of the ACM Conference on Computer and Communications Security (CCS 2012), New York, NY, USA, pp. 217–228. ACM (2012)
7. Backes, M., Gerling, S., Hammer, C., Maffei, M., von Styp-Rekowsky, P.: AppGuard – enforcing user requirements on android apps. In: Piterman, N., Smolka, S.A. (eds.) TACAS 2013 (ETAPS 2013). LNCS, vol. 7795, pp. 543–548. Springer, Heidelberg (2013)
8. Barrera, D., Clark, J., McCarney, D., van Oorschot, P.C.: Understanding and improving app installation security mechanisms through empirical analysis of android. In: SPSMD (SPSM 2012), New York, NY, USA, pp. 81–92. ACM (2012)
9. Beresford, A.R., Rice, A., Skehin, N., Sohan, R.: Mockdroid: trading privacy for application functionality on smartphones. In: HotMobile 2011, pp. 49–54 (2011)
10. Enck, W., Ongtang, M., McDaniel, P.D., et al.: Understanding android security. IEEE Secur. Priv. **7**(1), 50–57 (2009)
11. Felt, A.P., Chin, E., Hanna, S., Song, D., Wagner, D.: Android permissions demystified. In: 18th CCS, pp. 627–638. ACM (2011)

12. Felt, A.P., Ha, E., Egelman, S., Haney, A., Chin, E., Wagner, D.: Android permissions : user attention, comprehension, and behavior. In: SOUPS 2012, New York, NY, USA, pp. 3:1–3:14. ACM (2012)
13. Guha, S., Jain, M., Padmanabhan, V.N.: Koi: a location-privacy platform for smartphone apps. In: NSDI, NSDI 2012, p. 14. USENIX Association (2012)
14. Hildenbrand, J.: Android app permissions - how google gets it right. http://www.windowsphone.com/
15. Hornyack, P., Han, S., Jung, J., Schechter, S., Wetherall, D.: These aren't the droids you're looking for: retrofitting android to protect data from imperious applications. In: CCS (2011)
16. Ismail, Q., Ahmed, T., Kapadia, A., Reiter, M.K.: Crowdsourced exploration of security configurations. In: Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI 2015), pp. 467–476, New York, NY, USA. ACM (2015)
17. University of Alabama at Birmingham Online. The future of mobile application. http://businessdegrees.uab.edu/resources/infographics/the-future-of-mobile-application/
18. Rashidi, B., Fung, C., Dude, T.: Ask the experts! android resource access permission recommendation with recDroid. In: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 296–304, May 2015
19. Rashidi, B., Fung, C., Vu, T.: Android fine-grained permission control system with real-time expert recommendations. Pervasive Mob. Comput. (2016)
20. Russello, G., Jimenez, A.B., Naderi, H., van der Mark, W.: Firedroid: hardening security in almost-stock android. In: ACSAC 2013, New York, NY, USA, pp. 319–328. ACM (2013)
21. Shabtai, A., Fledel, Y., Kanonov, U., Elovici, Y., Dolev, S., Glezer, C.: Google android: a comprehensive security assessment. IEEE Secur. Priv. **8**(2), 35–44 (2010)
22. Victor, H.: Android's google play beats app store with over 1 million apps, now officially largest. http://www.phonearena.com/news/Androids-Google-Play-beats-App-Store-with-over-1-million-apps-now-officially-largest_id45680
23. Xu, R., Sadi, H., Anderson, R.: Aurasium: practical policy enforcement for android applications. In: 21st CSS, Security 2012, p. 27. USENIX Association (2012)
24. Yang, L., Boushehrinejadmoradi, N., Roy, P., Ganapathy, V., Iftode, L.: Short paper : enhancing users' comprehension of android permissions. In: Proceedings of the Second ACM Workshop on Security and Privacy in Smartphones and Mobile Devices (SPSM 2012), New York, NY, USA, pp. 21–26. ACM (2012)
25. Zhou, Y., Zhang, X., Jiang, X., Freeh, V.W.: Taming information-stealing smartphone applications (on android). In: McCune, J.M., Balacheff, B., Perrig, A., Sadeghi, A.-R., Sasse, A., Beres, Y. (eds.) Trust 2011. LNCS, vol. 6740, pp. 93–107. Springer, Heidelberg (2011)