2019

The Evaluation of an Android Permission Management System Based on Crowdsourcing

Pulkit Rustgi
Virginia Commonwealth University

THE EVALUATION OF AN ANDROID PERMISSION MANAGEMENT SYSTEM BASED ON CROWDSOURCING

A Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University.

by

PULKIT RUSTGI

Bachelor of Science from Virginia Commonwealth University, 2017

Director: Dr. Carol Fung,

Assistant Professor, Department of Computer Science

Virginia Commonwewalth University

Richmond, Virginia

August, 2019

Acknowledgements

I would like to thank my advisor Dr. Carol Fung for giving me an opportunity and countless hours of her time. In addition, I would also like to thank my committee members, Dr. Eyuphan Bulut and Dr. Carl Elks, for the assistance they have provided as well as the time they have taken out of their schedules to validate this work. Lastly, I'd like to thank my family and friends who encouraged me to further my education and seek out greater knowledge.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract

THE EVALUATION OF AN ANDROID PERMISSION MANAGEMENT SYSTEM BASED ON CROWDSOURCING

By Pulkit Rustgi

A Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science at Virginia Commonwealth University.

Virginia Commonwealth University, 2019.

Director: Dr. Carol Fung,

Assistant Professor, Department of Computer Science

Mobile and web application security, particularly concerning the area of data privacy, has received much attention from the public in recent years. Most applications are installed without disclosing full information to users and clearly stating what they have access to. This often raises concerns when users become aware of unnecessary information being collected or stored. Unfortunately, most users have little to no technical knowledge in regard to what permissions should be granted and can only rely on their intuition and past experiences to make relatively uninformed decisions.

DroidNet, a crowdsource based Android recommendation tool and framework, is a proposed avenue for the technically incapable. DroidNet alleviates privacy concerns and presents users with permission recommendations of high confidence based on the decisions from expert users on the network who are using the same applications. The framework combines an interactive user interface, used for data collection and presenting permission recommendations to users, with a transitional Bayesian inference

model and multiple algorithms used for rating users based on their respective expertise levels. As a result, the recommendations that are provided to users are based on aggregated expert responses and their confidence levels.

This work presents the completed DroidNet project in its entirety, including the implementation of the application, algorithms, and user interface itself. Additionally, this thesis presents and utilizes a unique collection of real-world data from actual Android users. The primary goal of this work is to evaluate the effectiveness and accuracy of DroidNet's recommendations and to show that regular mobile device users can benefit from crowdsourcing.

CHAPTER 1

INTRODUCTION

Mobile applications, or apps for short, have become a part of most individuals day to day lives. Whether it be dealing with business, social interaction, or marketing prospects, apps are ubiquitous. According to a report, the Google Play Store, which houses most Android apps, first surpassed 1 million unique apps available for download in July 2013. Currently, as of June 2019, the store has over 2.7 million apps available for download [1] which shows mobile development's unparalleled rise over the past decade.

With such a large user base that only appears to be broadening, concerns with privacy and security only seem to be growing. Due to the nature and rate of growth for marketplaces housing these apps, such as the aforementioned Google Play Store and its competitor the Apple App Store, it seems nearly impossible to verify the true intention of an application before it is uploaded to a market. As a result, this means that the current permission management for privacy control on the Android operating system (OS) heavily depends on users. Users have to manually decide what permissions an app can access after an app is installed. Apps are free to request users give access to whatever resources they claim to need, as most of the time these requests are not validated nor the developers held responsible.

Because of the lack of reliable verification processes, other than developers apps being checked manually, up to 70% of apps request access to permissions that they do not necessarily even require [2]. This translates to untrustworthy apps with hidden agendas often being downloaded by unsuspecting users. These malicious *third-party* apps not only steal private information, such as online credentials, private messages, contacts, and the user's live location, but also cause the victim to experience monetary loss. In rare cases, these malicious apps even utilize users' devices to send text messages as well as paid phone calls [3].

Permissions requested by these, or any, apps are often displayed to users through a dialog window with little information other than their name being disclosed. This has proved to be ineffective since users may not entirely understand what resource an application is requesting access to and/or they have no time or incentive to look into what granting or denying a specific request may cause [4, 5]. Even then, only a small percentage (3%) of users correctly respond to requests that relate to permissions [2].

Thus, less proficient users often turn to more seasoned experts with a greater understanding of a particular field or application for questions concerning settings relating to permissions. DroidNet[6], proposed by Rashidi et al., is a solution that helps users alleviate their privacy and security concerns through seeking opinions from expert users. DroidNet is a permission control system that examines individual users' app permission settings and compares them with recommended decisions through crowdsourcing. When mismatches between permissions and recommendations from the framework are found, DroidNet notifies users on its live network through Android's notification system so that they can potentially revise their decisions. The recommendations are crowdsourced, meaning they rely on both verified expert users as well as highly rated normal users who use the same apps.

The tool also allows users to easily view permissions requested by each individual app, in much more efficient manner than Android's current design, as well as manage them utilizing a simple system framework that provides the following major functionalities:

- A front-end user interface and launcher pages to indicate which apps DroidNet is currently monitoring.
- A back-end engine to collect each users' installed application and its corresponding permissions and responses.
- A recommendation system and display to guide users.
- A user-based ranking algorithm to rank security risks of each mobile application as well as their permissions.

DroidNet is a live tool available on the Android market, through the Google Play Store, that provides users with secure permission recommendations for apps installed on their device. This work describes the process behind the development of the application itself, the implementation of the back-end algorithms associated with the application used for ratings and recommendations, and the evaluation of how practical and reliable the system is for the average users on the network using a real-world dataset.

1.1 Summary of Contribution

The contributions of this work are as follows:

- Implementation of the Android DroidNet application.
- Collection of unique dataset containing users as well as their respective apps and permissions.
- Implementation of multiple algorithms associated with DroidNet.
- Unique recommendations tailored to any current and future DroidNet users.

CHAPTER 2

BACKGROUND

Crowdsourcing is defined as a distributed online process that often involves the Internet and the participation of a crowd [7]. It is often described as an outsourcing process involving data collection done through tools, such as Google Forms or Amazon Mechanical Turk, where users complete a predefined task. These surveys are often aimed at the collection of data enclosed within a small subset of the general population that still have key overlapping factors or characteristics. This overlapping factor can be one of many data points, such as demographic information, or even the participation in a common activity. The only key aspect of this factor is that it often shapes the dataset and is generally the initial motivation behind the data collection itself it.

This chapter briefly overviews Android's native architecture and general design. First, it details permissions, and what they are, in the context of Android apps. Next, an in-depth discussion takes place on how various permissions are generated by developers, what permissions are presented as requests to users, and what specific types of permissions, and their associated groups, mean. Then, management and requests in relation to permissions are discussed.

Following the discussion of permission management and request handling, other existing mobile privacy solutions are covered in order to understand the type and extent of work done similar to DroidNet. Lastly, this chapter introduces DroidNet and its currently implemented rating model for users. In addition to this, it includes the changes made to the initial DroidNet design in order to refine results as well as accommodate the Android OS and, as aforementioned, its permission management and recommendation system.

2.1 Permission Types and Classification

Android utilizes various categories of permissions that are split into larger groups. Android refers to these groups as **permission groups**. These permission groups contain individual permissions that fit a particular group or category as deemed appropriate by Android. Application developers may place various Application Programming Interface (API) calls into their application to initialize these permission requests and prompt the user to take action. A full example of the process can be seen in figure 3.

According to the Android developer portal, if an application requests even a single permission through the API that falls into one of these permission groups the whole permission group is either denied or granted depending on the user's answer to the request [8]. This behavior is particularly troublesome as only the group permission is shown to the user rather than getting into the specifics of which exact permission the application requires access to.

What's even more concerning is when an application requests access to a particular "dangerous permission", or permissions that grants access to sensitive information on your device, such as contact lists. Due to the aforementioned behavior of permission groups, other permissions falling into the same group no longer need to request access from the user in any future cases, regardless of how relevant the permissions are to the application [8].

This paper examines ten out of twelve possible Android permissions. They are as follows, with definitions as provided through the developers of Android themselves:

- **Storage Access**: Used for runtime permissions related to the shared external storage.

- **Location Access**: Used for permissions that allow accessing the device location.
- **Contact Access**: Used for runtime permissions related to contacts and profiles on this device.
- **Phone Access**: Used for permissions that are associated telephony features.
- **Camera Access**: Used for permissions that are associated with accessing camera or capturing images/video from the device.
- **Microphone Access**: Used for permissions that are associated with accessing microphone audio from the device.
- **In-App Purchases**: In-App Purchases (or IAPs) are available to be made in the app or game using Google Play In-app Billing.
- **SMS Access**: Used for runtime permissions related to user's SMS messages.
- **Calendar Access**: Used for runtime permissions related to a user's calendar.
- **Sensors**: Used for permissions that are associated with accessing body or environmental sensors.

Activity Recognition as well as *Call Logs* are the remaining two permissions, but they were not analyzed due to a lack of available results or apps utilizing them. This would mean that any recommendation being made would inherently lack credibility.

2.2 Permission Management and Requests in Android

Most Android warnings relating to permissions are not taken into account by users [9]. Ultimately, many warnings have little to no impact on their decisions and alternative solutions, other than the currently implemented interface dialogs created by the native Android OS environment warning users, are needed. However, while

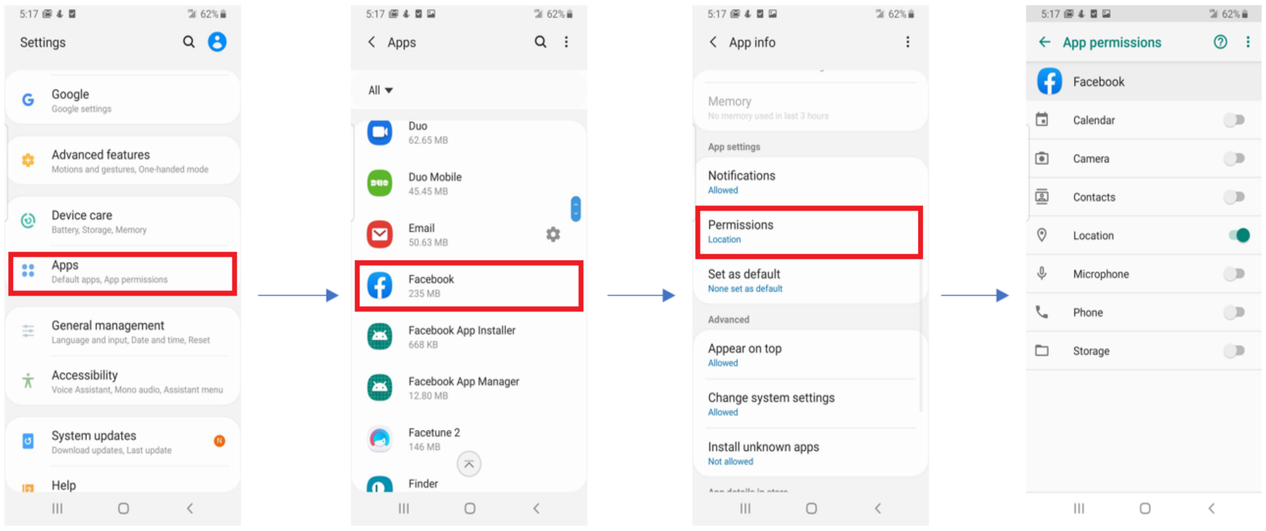


Fig. 1.: Manual Process for Modifying App Permissions

many attempts to create alternatives have been attempted, ultimately the system must be tested as well as shown to be trustworthy to the regular mobile user who lacks the technical knowledge to make informed decisions.

While third-party systems may need validation and extensive testing, the Android system for permission management periodically goes through changes and updates which may not be straightforward for users. Additionally, the current user interface (UI) is cluttered and involves unnecessary steps to access specific system pages which the average user may not be aware of. User interfaces are also dependent on a phone's manufacturer, and the current version of an OS heavily dictates the design scheme they follow. These areas may potentially cause a sense of confusion in a user when first navigating through a new device or when updating to a new OS version.

Most importantly, as discussed in the following subsection, users are not given the opportunity to make there own choices upon an app's initial install.

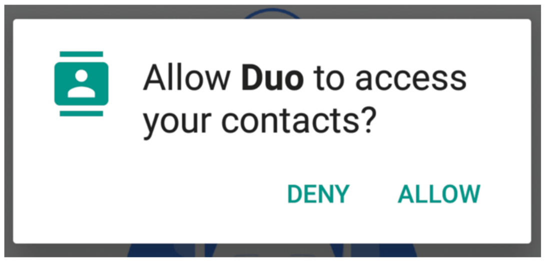


Fig. 2.: Example Android Permission Request

2.2.1 Permission Management

Most mobile apps come with their own individual permissions once downloaded. Each application on Android, downloaded from the Google Play Store, initially comes with all of its permissions denied, or turned off. If a user ever wants to change an app's permissions, they must go through the Android "Settings" menu, select which application they want to alter, and then modify the settings individually for each permission (as shown in figure 1). The issue with this system design is that users are disincentivized by the process taking far too long. To users, the intrinsic cost, on the surface, for modifying each app appears to not be worth it. They simply want to use the application and aren't concerned with the consequences that may occur later.

2.2.2 Permission Requests

Permission requests have changed over time, specifically in the fashion they are presented. Android versions below 6.0, codename "Marshmallow", granted an application access to all requested permissions at installation time. Currently, and in any version above 6.0, permission requests are displayed on-screen once a user installs an application and opens it on their device (see figure 2 for an example).

These requests are situational and are triggered by predefined events relating to the app's behavior, but are called by specific in-device actions. Calls for requests

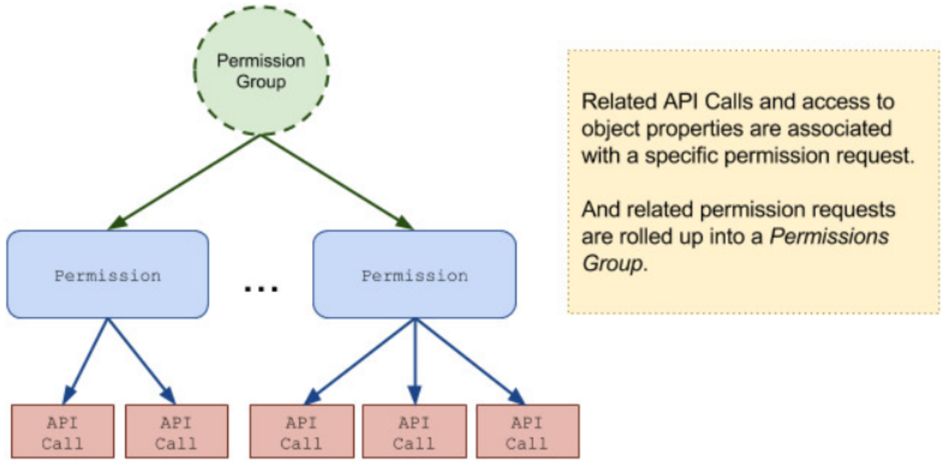

Permission Group
Permission
...
Permission
API Call
API Call
API Call
API Call
API Call
Related API Calls and access to object properties are associated with a specific permission request.
And related permission requests are rolled up into a *Permissions Group*.

Fig. 3.: Permission Request Flow [8]

are generated by the developer meaning apps may or may not necessarily require the requested permission(s) to function. Alternatively, an application may on load up be designed to request a various set of permissions by the developer through placing specific API calls targeted at an application being opened for the first time.

2.3 Existing Mobile Privacy Solutions

Multiple attempts to alleviate concerns with mobile application privacy have taken place. However, previously proposed solutions, as detailed in this section, all lack the idea of including users' expertise levels when making recommendations.

RecDroid, proposed by Rashidi et al. [10], was a previous iteration and the initial idea behind DroidNet. RecDroid functioned by allowing apps to be installed in two different modes: (1) trusted mode; (2) probation mode. At run time, if the installed app is run on probation mode it is required to request permissions from users when attempting to access sensitive resources. In trusted mode, the app in question is not required to request any permissions as all permission requests are considered trusted. However, the idea was simply not feasible as RecDroid required modifications to be made to the Android OS.

Android v4.3 briefly had a feature that allowed for users to disable permissions and their requests for selected apps on their devices. This feature, named App Opps [11], was labeled as experimental and removed due to the fact that individual permissions that were required by apps to fully function could be disabled causing an app to malfunction and crash.

Following the principal of least privilege, Lin et al. similarly propose permission management selecting a set of minimal permissions that allowed an application to keep its intended functionality [12]. The analysis for the proposed recommendations was done through a static code analysis tool named Androgaurd which looked at a set of user privacy profiles.

Similarly, a technique for selecting a minimal set of permissions for an application to use is proposed by Ismail et al. [13]. However, the solution they propose repackages apps with a pre-defined set of permissions, creating an unreasonable amount of combinations. The primary fault with this implementation is the idea of an application being repackaged in order to create an unknown amount of feasible permission combinations.

Gort [14], a semantic based solution proposed by Shahriyar, analyzes an app's behavior while also examining the context and semantics of the app's description(s). Workers are also asked to examine an application and decide whether a permission request makes sense, and if so whether that request should be granted or denied.

Liu et al. propose PriWe [15], which identifies permission requests based on users' expectations. Users with similar responses to a set of permission requests are looked at, and similar users are identified and shown each others setting profiles.

Yang et al. suggested the idea of a system for users to be able to share their permission recommendations with others [16]. They proposed that users leave comments

Notation	Description
$\mathcal{U}$	$\{U_1, \ldots, U_n\}$: 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.

Table 1.: DroidNet Notations

or reviews for specific apps and that these reviews be ranked in order to display a set of top reviews that represent high quality settings for apps.

Lastly, and most similar to DroidNet other than RecDroid, Agarwal et al. propose PMP [17]. PMP collects privacy data and uses *majority voting* to recommend settings to other iOS users. While it may collect application data similar to DroidNet, PMP results in a high false recommendation rate due to using simple majority voting instead of weighted voting [18].

2.4 DroidNet Network

The lemmas and proofs for the algorithms and equations included in this section can be found in the original DroidNet paper [6] and will not be repeated in this section. Rather, this section will discuss the surface level knowledge and terminology needed to understand how the app's design, rating system, and recommendation system were developed in order to evaluate it. Table 1 shows the notations provided by the original paper.

This section begins by introducing and defining the following user types on the network:

- Seed Experts

- Regular Users
    1. Direct Users
    2. Indirect Users
    3. Disconnected Users

2.4.1 Seed Experts

Seed experts (SE) are users that are preselected as having advanced knowledge relating to one or more apps being utilized for ratings. Seed experts are given a *rating* of **1.0**, which is the highest possible rating on the network, and every permission decision they have made is assumed to be correct. Seed experts are also used to begin the initial rating of other users on the network, beginning with users who directly share one or more overlapping apps with them. The set of permission settings/recommendations made by seed experts is referred to as the *ground truth* (GT), and the apps associated with these permissions ground truth apps respectively.

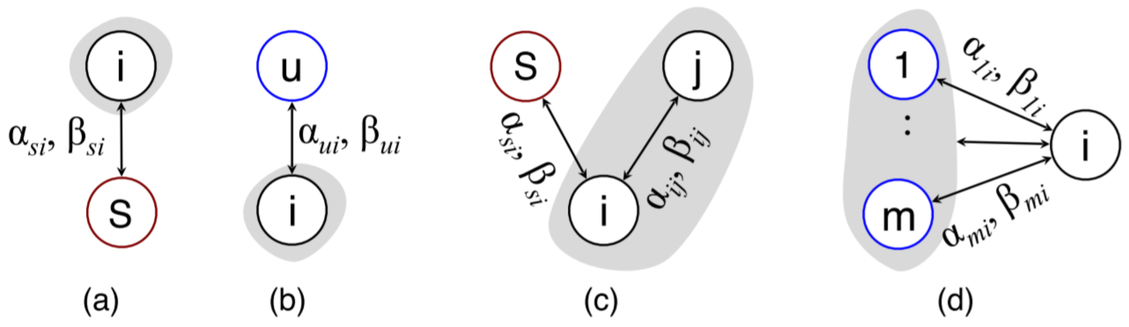

i
α_{si}, β_{si}
S
(a)
u
α_{ui}, β_{ui}
i
(b)
S
j
α_{si}, β_{si}
α_{ij}, β_{ij}
i
(c)
1
α_{1i}, β_{1i}
:
i
m
α_{mi}, β_{mi}
(d)

Fig. 4.: DroidNet Graph Cases Between Users [6]

2.4.2 Direct Users

A *direct user* is defined as a neighbor, or a user on the network that shares an application overlap exceeding threshold θ (where θ is generally equal to one), of a seed expert (see figure 4a). Initially, each user overlapped with a seed expert has its permissions compared for matches and mismatches. Once the initial set of comparisons is made, and the similarity tuple of permissions for the current user calculated, both the rating and confidence of said user can be found. Additionally, direct users participate in the rating of indirect users as discussed in the next subsection.

2.4.3 Indirect Users

Indirect users are users not connected directly to a seed expert, but share a common overlap with a rated direct user (see figure 4c). Two users become indirect neighbors if they share an application overlap with a third intermediary user. Indirect users are able to participate in the rating of other unrated users as long as they share an application overlap with them. Users assigned ratings through other indirect users are also referred to as indirect users.

2.4.4 Disconnected Users

Disconnected users are can be defined as regular users who have no neighbor or application overlap on the network, thus making them unreachable in any graph. These users are unable to be rated or participate in the permission recommendation process. Additionally, the apps they provide are not included in the pools set of potentially rated apps, as they share no overlap to provide sufficient recommendation verification.

2.4.5 User Expertise Rating

A *rating* is defined as a classification or ranking of someone or something based on a comparative assessment of their quality, standard, or performance. DroidNet functions by assigning each user on its network a rating, or *expertise level.* Expertise level can be defined as the probability that a user makes the correct decision when granting or denying a permission. To calculate the total expertise level of user i each overlapped application must have their permissions compared to either a seed expert or an expert user, and the total number of matching and mismatching permissions calculated and used for a final rating. This set of comparisons can be referred to as the *similarity* between two users, and is defined in the paper as the following tuple shown in equation 2.1 and 2.2:

$$\begin{aligned}\alpha_{ij}^{(n)} &= \sum_{k=1}^{n} q^{n-k} x_k + q^n C_0 \\ &= x_n + q x_{n-1} + \ldots + q^{n-1} x_1 + q^n C_0\end{aligned} \tag{2.1}$$

$$\begin{aligned}\beta_{ij}^{(n)} &= \sum_{k=1}^{n} q^{n-k} (1 - x_k) + q^n C_0 \\ &= (1 - x_n) + q(1 - x_{n-1}) + \ldots + q^{n-1}(1 - x_1) + q^n C_0\end{aligned} \tag{2.2}$$

In these equations, $q \epsilon [0, 1]$ refers to a remembering factor that's used to discount past observations which in turn puts an emphases on more recent permissions and recommendations. C_0 refers to a constant weighting of the *initial belief.* The initial belief is used to make sure that an unrated user, or child user, remains a lower score than the neighbor(s) that went into its rating calculation, or parent(s). The weight of this constant is used in every final rating calculation for direct and indirect users. This means that even if a user matches its parent neighbor(s) entirely in terms of permission, meaning every grant and deny request is identical, said user will still have a slightly reduced score.

$$R_i = \frac{\alpha_{si}}{\alpha_{si} + \beta_{si}} \tag{2.3}$$

$$C_i = 1 - \sqrt{\frac{12\alpha_{si}\beta_{si}}{(\alpha_{si} + \beta_{si})^2(\alpha_{si} + \beta_{si} + 1)}} \tag{2.4}$$

The rating of a direct user is done by comparing the number of matching permissions over the total number of permissions for all overlapped seed experts while using a constant value of the initial belief for normalization. Equations 2.3 and 2.4 respectively show how rating and confidence scores are calculated for a direct user i connected to one seed expert using a similarity tuple.

$$\alpha_i = \frac{\alpha_{ui} p_i p_u}{p_i p_u + (1 - p_i)(1 - p_u)} + \frac{\beta_{ui} p_i (1 - p_u)}{p_i(1 - p_u) + (1 - p_i) p_u} \tag{2.5}$$

$$\beta_i = \frac{\alpha_{ui} (1 - p_i)(1 - p_u)}{p_i p_u + (1 - p_i)(1 - p_u)} + \frac{\beta_{ui} p_u (1 - p_i)}{p_i(1 - p_u) + (1 - p_i) p_u} \tag{2.6}$$

When calculating ratings for indirect users, the existing rated users participating in the propagation must surpass a minimum rating threshold. The *rating threshold* (RT) is an arbitrary threshold placed to filter already rated users. This threshold is primarily utilized in calculations done for indirect neighbors of a seed expert, as direct neighbors of seed experts receive ratings from them without indirect influence. The expertise level of indirect users can be calculated using the formulas seen in equation 2.5 and 2.6 where p_i and p_u refer to the true/known expertise levels of user i and u as described in Table 1.

$$\begin{aligned} \alpha_i &= \alpha_i^1 + \ldots + \alpha_i^m = \sum_{k=1}^{m} \alpha_i^k \\ \beta_i &= \beta_i^1 + \ldots + \beta_i^m = \sum_{k=1}^{m} \beta_i^k \end{aligned} \tag{2.7}$$

$$\begin{aligned} \alpha_i^k &= \alpha_{ki} \frac{\alpha_k \alpha_i}{\alpha_k \alpha_i + \beta_k \beta_i} + \beta_{ki} \frac{\beta_k \alpha_i}{\beta_k \alpha_i + \alpha_k \beta_i}, \forall k \in \{1, 2, \ldots, m\} \\ \beta_i^k &= \alpha_{ki} \frac{\beta_k \beta_i}{\alpha_k \alpha_i + \beta_k \beta_i} + \beta_{ki} \frac{\alpha_k \beta_i}{\beta_k \alpha_i + \alpha_k \beta_i}, \forall i \in \{1, 2, \ldots, m\} \end{aligned} \tag{2.8}$$

However, a temporary placeholder value must be used to initially estimate p_i as it is unknown (and currently being calculated). The intuitive choice is to select the assumed expertise in the initial round of every user, with round and initial round being denoted as t and t^0 respectively, to be $\frac{1}{2}$ as that is considered "average" expertise. Once that placeholder value is selected, the values for a_i and β_i are calculated iteratively by replacing p_i with the previous rounds expertise level $R_i^{(\text{t-1})}$. This is done until the values converge, or fall below a defined threshold when subtracting the previous round from the current rounds value. This process is done for every rated neighbor that a user has that falls above a rating threshold (as described later). The user i's final score is calculated as $R_i = a_i/(a_i + \beta_i)$ where a_i and β_i are the summation of all the previously calculated neighbors a's and β's respectively in the iterative step above (shown in equations 2.7 and 2.8).

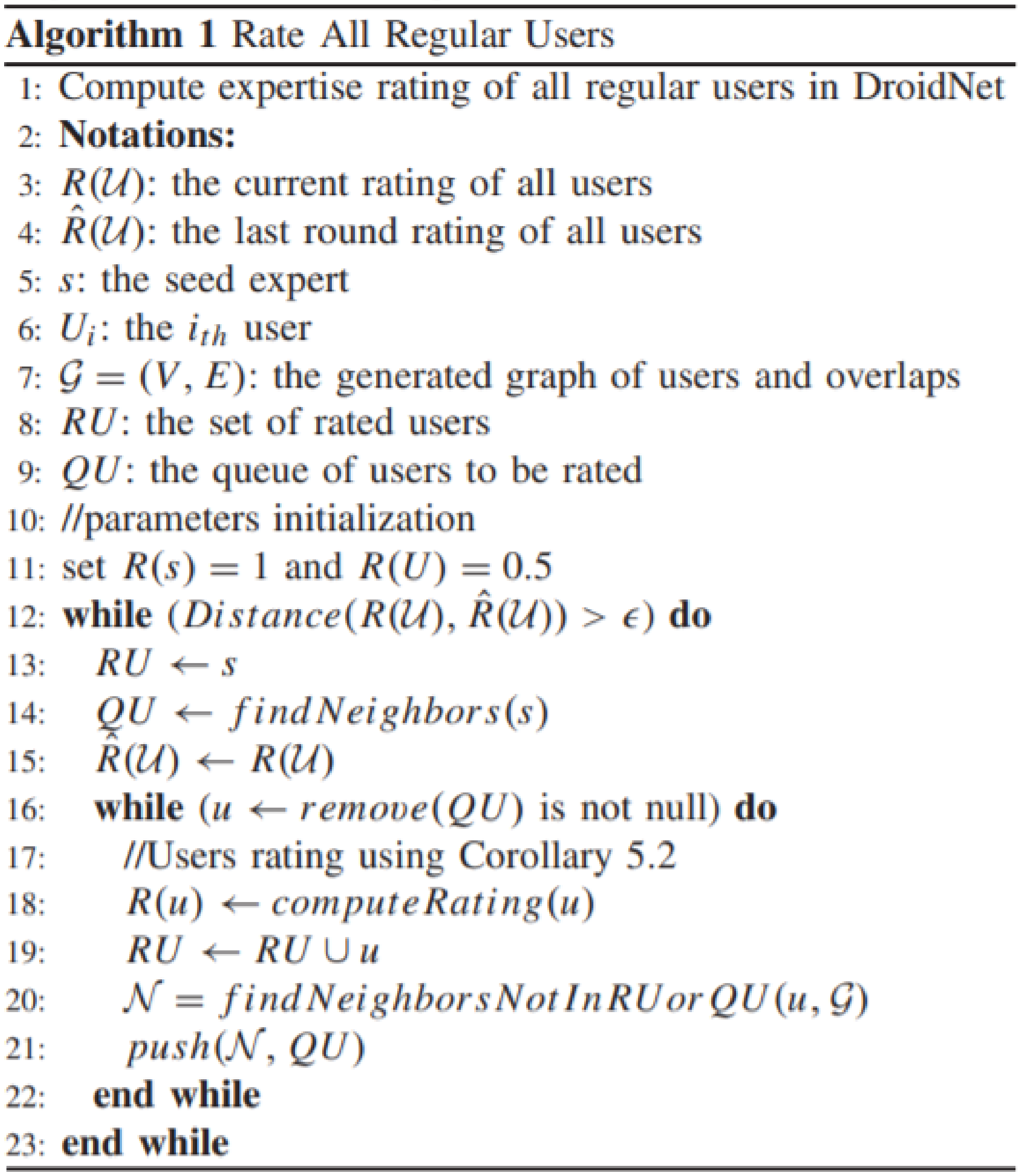
Essentially, the rating equation of a user with neighbors can be simplified to the number of correctly matching responses plus a constant C_0 divided by the total number of overlapping permissions plus two times C_0. As a reminder, only users with sufficient ratings, exceeding a predefined RT, and confidence levels, denoted as C, are used in rating calculations involving other unrated users.

2.4.6 Estimating User Ratings

The requirements for a user to be rated depend on three factors: (1) the round the user is rated on; (2) the constant factor C_0 used for value normalization in calculations; and (3) the RT. Here, a *round* refers to the particular order and iteration that a rating takes place. The amount of rounds that take place are dependent on the quantity, quality, and diversity of the data being analyzed. However, rounds are consistent in the order they take occur. The general structure of user rating relating to rounds is as follows:

1. Round 1: Seed Experts receive ratings.
2. Round 2: Direct Users receive ratings.
3. Round 3 - Second To Last Round: Indirect Users receive ratings.
4. Final Round: Users unable to be rated are classified as unratable indicated by a negative rating.

For clarity, figure 5 shows the full algorithm, **Algorithm 1**, from the original paper.



```
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
```

Fig. 5.: DroidNet Rate All Regular Users Algorithm [6]

2.4.7 Providing Permission Recommendations for Applications

Once every user receives their rating from the direct and indirect rating methods, or they are unable to be rated, the network is capable of recommending permissions for each available application in its existing pool. As described in the original paper, the algorithm for recommendations requires the following:

1. The application having its permission rated.
2. The set of users above the expertise threshold τ_e with the application installed.
3. The maximum and minimum threshold to consider granting a permission τ_d.

Similar to the RT, the *expertise threshold*, denoted as τ_e, is a threshold used for filtering out users with insufficient rating. τ_e is selected arbitrarily, but should be kept on the higher end due to the value representing what the network considers a sufficient enough score for any rated user to be seen as an expert. As the idea behind weighted voting is for a larger emphasis to be placed on higher rated users, result accuracy should increase with reduced participation from less inexperienced users.

The maximum and minimum decision thresholds are thresholds used for the weighted voting recommendation system and its associated algorithm. The algorithm, labeled **Algorithm 2** in the original paper, is shown in figure 6. The threshold for permission recommendation can be denoted as τ_d. As calculated in the algorithm, a permission recommendation depends on its aggregate score, which can be calculated as $\frac{\alpha}{\alpha+\beta}$. Here, α and β represent the summation of the total number of users who voted yes, meaning grant, or, meaning deny, that app's permission respectively. It is important to note that only users who exceed the τ_e are used in recommendations.

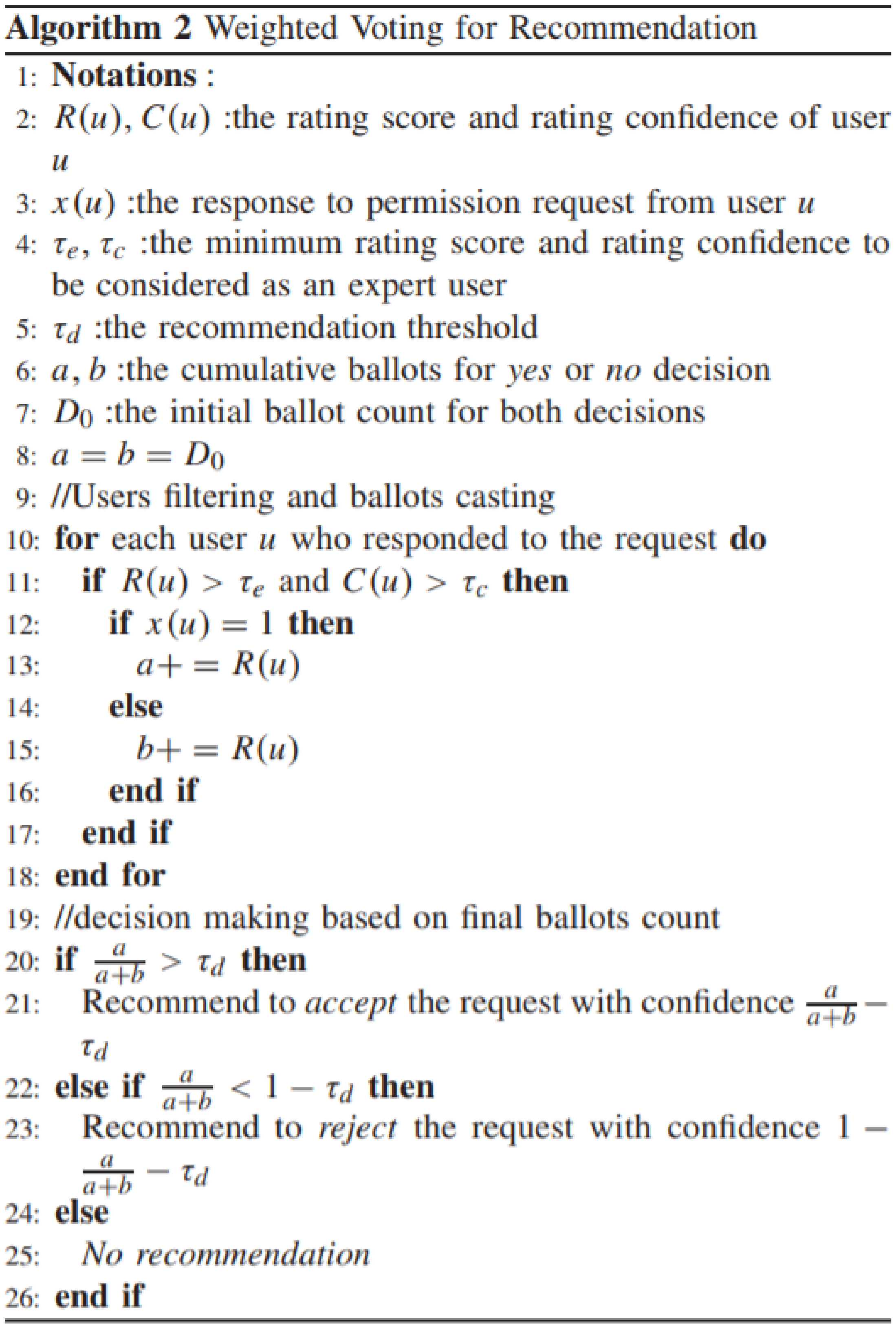
Once the aggregate score is calculated, a permission recommendation can take any of the three following cases:

- Grant
- Deny
- No Recommendation Available (NRA) due to insufficient scoring

Case 1, the recommendation to grant a permission, occurs when the aggregate score of a permission falls above the required threshold τ_d. Case 2, the recommendation to deny a permission, occurs when the score of a permission falls below the threshold value 1 - τ_d. Case 3 occurs when the score of a permission falls in between the upper and lower thresholds of τ_d and 1 - τ_d. In each of these cases, the confidence at which the recommendation is made is represented by the calculated score.

Three classifications for confidence level, high, medium, and low, are used in order to communicate to DroidNet users the internal confidence of the recommendation without detailing the intricacies of the defined threshold(s). The first level of "high" occurs when a permissions confidence level is greater than 0.66. A "medium" recommendation occurs when a confidence level falls between 0.66 and 0.33. Lastly, a low recommendation occurs when confidence falls below 0.33. DroidNet aims at high confidence recommendations, but depending on the selected value of τ_d the other two cases may occur at low rates.

The full implementation and design of these algorithms can be found in the following chapter.



```
Algorithm 2 Weighted Voting for Recommendation
 1: Notations :
 2: $R(u), C(u)$ :the rating score and rating 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
```

Fig. 6.: DroidNet Weighted Voting for Recommendation Algorithm [6]

CHAPTER 3

DROIDNET IMPLEMENTATION AND DESIGN

This chapter overviews the general infrastructure of the DroidNet framework, as shown in figure 7. Specifically, this section discusses DroidNet's proposed implementation in the original paper and how the live implementation of DroidNet differs. This chapter begins with describing the implementation of the project in terms of specifications and design. It then details each of the individual DroidNet components and their purposes. Finally, a description of the original proposed recommendation algorithms as well as the modified algorithms are presented.

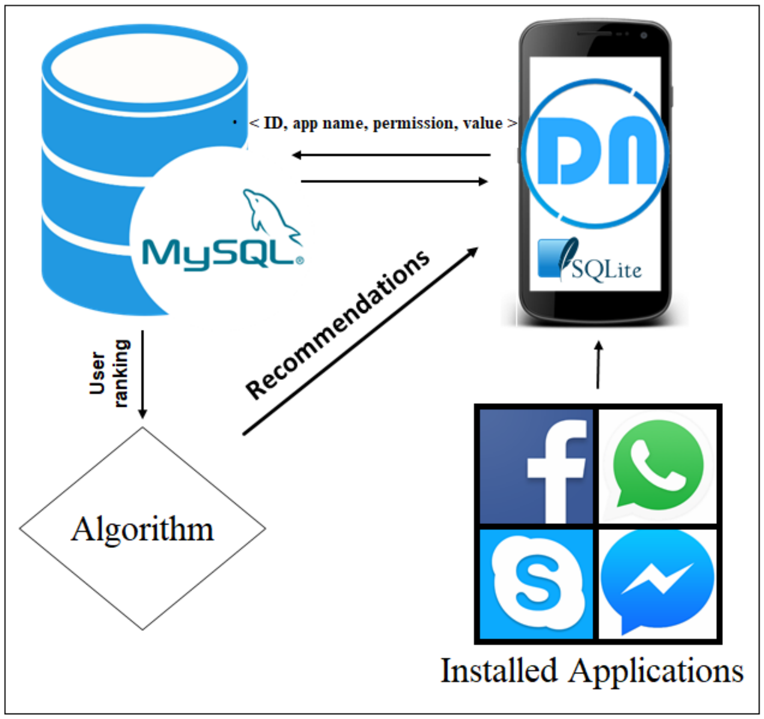


Fig. 7.: DroidNet Architecture

3.1 Tools and Languages

DroidNet was developed in Android Studio and written in Java. Queries made when accessing DroidNet's associated database entries are done using SQL in conjunction with PHP. The Android Studio version the application was deployed on was 3.2.

The following are the complete set of tools and technologies used in its development:

- Android Studio
- phpMyAdmin
- Eclipse IDE
- MySQL
- SQLite

3.2 User Interface

The DroidNet interface is split up into four core sections, or pages. They are as follows:

- The registration and tutorial page(s)
- The launcher and main page of the application
- The permission management page
- The recommendations page for each individual user and their apps

This section, and the subsequent subsections, detail the purpose and general overview of each of the aforementioned pages. Each of the individual pages make up key components of the application. The flow from page to page is shown in figure 8. For the purposes of this paper a description of the DroidNet settings page will not

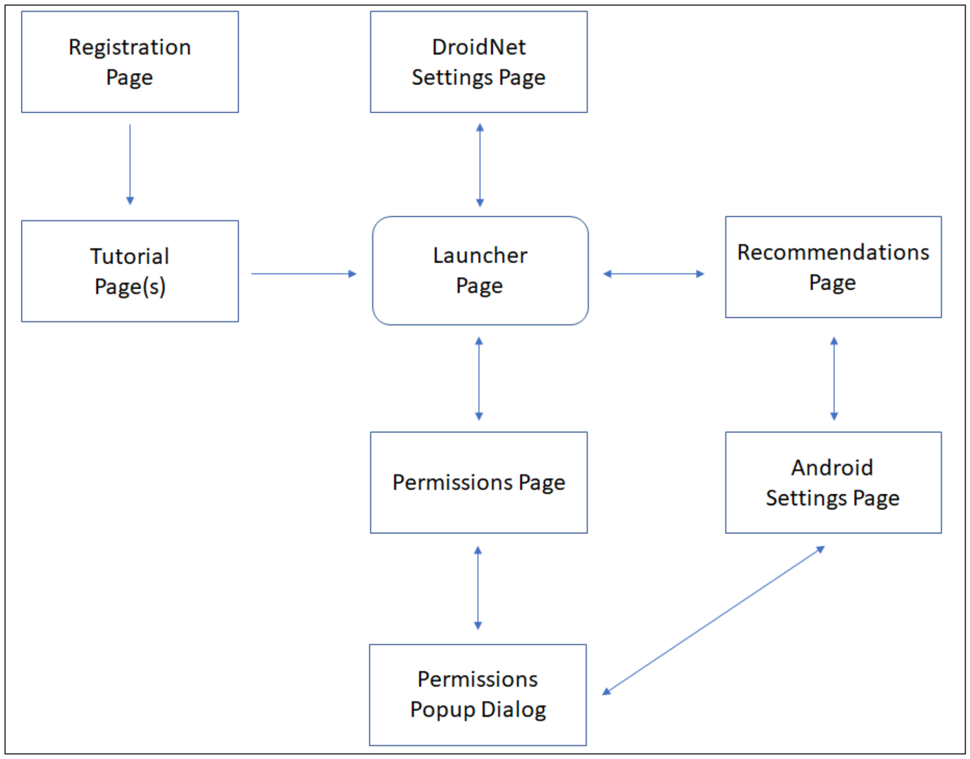


Fig. 8.: DroidNet Page Flow

be described, as its functionality is confined to the actual DroidNet application and was not required to be utilized for evaluation purposes.

3.2.1 Registration and Tutorial Pages

Upon initial install the user is initially presented with an End-User License Agreement (EULA) which must be accepted. The EULA is required for protection purposes, both for the user and the developer, as it allows for much needed transparency for those willing to read it. Users are then required to fill out a form-based page that is presented to them, as shown in figure 9. This page, and form, is primarily used for this works demographics, as users are later defined by *hashed and salted* device ID's rather than manual input for consistency. The form information users are required to fill for registration are as follows:

1. Email
2. Password
3. Password Confirmation
4. Age Group

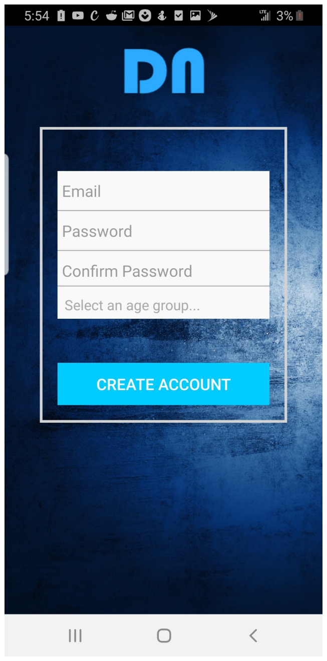
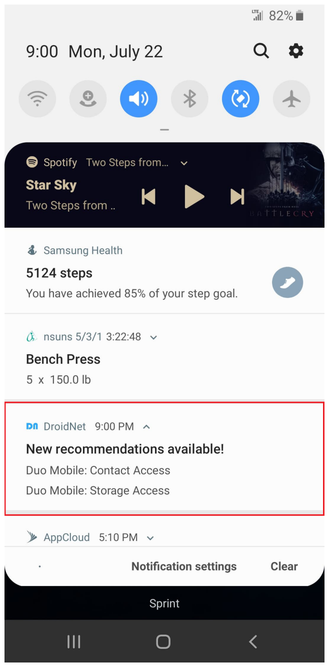


Fig. 9.: DroidNet Registration



Spotify Two Steps from...
Star Sky
Two Steps from ..

Samsung Health
5124 steps
You have achieved 85% of your step goal.

nsuns 5/3/1 3:22:48
Bench Press
5 x 150.0 lb

DroidNet 9:00 PM
New recommendations available!
Duo Mobile: Contact Access
Duo Mobile: Storage Access

AppCloud 5:10 PM

Fig. 10.: DroidNet Notification Alert

As the device ID can be used for identifying a user due to its uniqueness it is kept obscured from the network and never stored in plaintext. This applies to all users' passwords as well, which in addition is salted. Once the form is submitted the user can progress onto the tutorial pages. The app's tutorial consists of three pages which show in animated format the functionalities of the application and what the user can expect from it. After completing the tutorial the user is prompted to verify that they understood it, and if they agree they are then granted access to the full application itself.

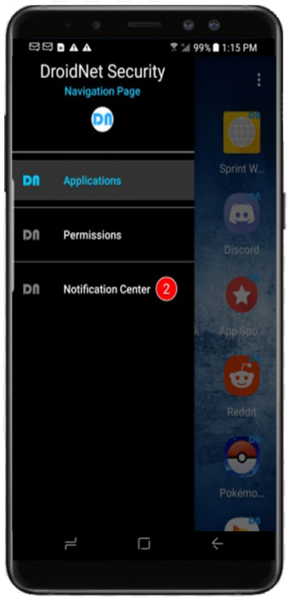


Fig. 11.: Launcher Page

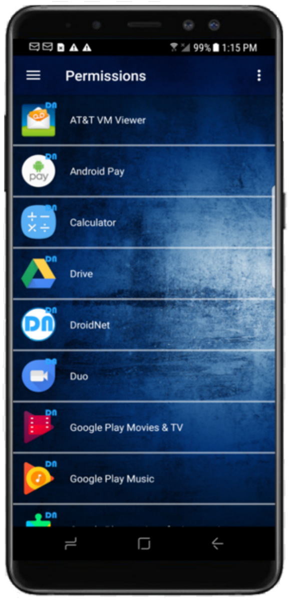


Fig. 12.: Permission Page

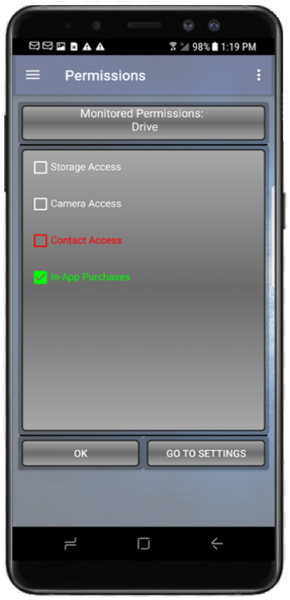


Fig. 13.: Permission Popup

3.2.2 Launcher Page

The *launcher page* functions as the *main* or front page of the application itself. The page works like the actual Android launcher page, but also as an indicator to the user as to what apps are being monitored by their DroidNet application. Each of the launcher item's images are replaced with an identical image with a small overlay containing the DroidNet DN logo, as shown in figure 11. Through here the user is able to navigate to the other parts of the application, such as the permissions, notifications, and settings page.

3.2.3 Permission Management Page

Figure 12 shows the *permission management page*, or *Permission Center*, which comes with two functionalities. To alleviate some of the aforementioned Android UI design problems, DroidNet provides the user with a modified management page that has the following functionalities:

1. A scrollable interface that functions as an intractable list of all downloaded applications.

2. A redirect to a popup overlay for each individual application when an application's respective list element is clicked, selected, or pressed on.

The primary interface functions much like Android's current implementation of the "Apps" page, as shown earlier in subsection 2.2.1 figure 1. Once an application in the list is clicked, a popup page displaying the current permission settings is displayed as an overlay. The unique aspect of the permissions page, other than its popup, is its ability to display permission matches and mismatches between the user and DroidNet's recommendations. As shown in figure 13, an app's permission is highlighted in green on the popup overlay if the user's decision matches DroidNet's recommendation for it and red if it does not. Otherwise, the permission remains highlighted in white indicating no recommendation is available. Additionally, the popup shows the confidence for each highlighted permission, regardless of its color, as long as a recommendation by the network exists.

If the user chooses to modify a permission and take into account DroidNet's suggestion, they are able to navigate through the permission popup by clicking the **"GO TO SETTINGS"** button. This will take them to their Android devices native permission managemnt page, as Android apps are not allowed to interfere with system settings and permission management directly. This process reduces the amount of steps compared to directly accessing the Android settings page directly and then finding an application in order to modify its settings. After the user arrives at the app's settings page, if the permission they navigated from has an alternative recommendation, an overlay popup dialog detailing DroidNet's recommendation and confidence is displayed as a reminder (see figure 15).

DroidNet Confidence and Color Intervals	
High	Green
Medium	Yellow
Low	Red

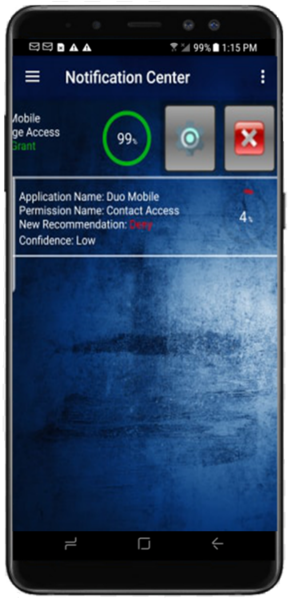
Table 2.: DroidNet Confidence Colors

Confidence classification, as discussed in subsection 2.4.7, determines the color selection of the circle displayed in the popup dialog. The colors associated with each confidence are displayed in Table 2.

3.2.4 Recommendation Page

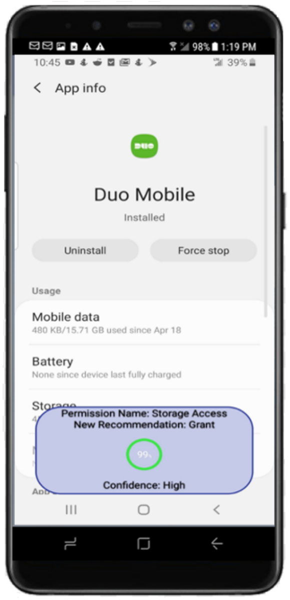
DroidNet's in app *recommendations page*, also referred to as the *notification center* (NC), houses all of its up to date recommendations for a users' mismatching app's permissions. Once the application is done loading it undergoes a *synchronization process*. This process, as seen in figure 16, is responsible for comparing the internal DroidNet database on the user's device, containing the permission entries of every application and the status of those permissions, to an identical table stored online. If mismatches are found between the two databases every online record is updated to match the offline record for a user.

After synchronization is complete, the users permission settings are compared to a table consisting of DroidNet's recommended permission settings. A flag is thrown alerting the user of mismatches and an entry of that mismatched permission is added to the list of permissions needing to be changed in the notification center. The insertion of new entries into the notification center creates a notification message that's displayed to the user through the Android notification bar. The notification, as seen in figure 10, contains information relating to the new recommendation, such



Application Name: Duo Mobile
Permission Name: Contact Access
New Recommendation: Deny
Confidence: Low

Fig. 14.: Notifications



Permission Name: Storage Access
New Recommendation: Grant
Confidence: High

Fig. 15.: Overlay Dialog

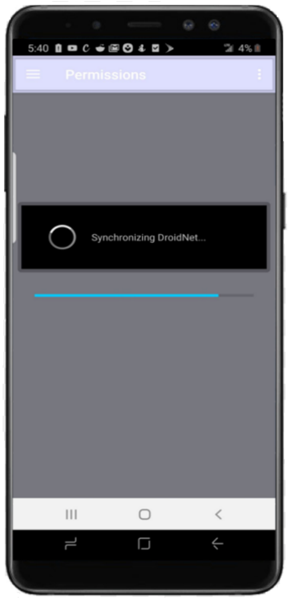


Fig. 16.: Synchronization

as the app name and advise permission setting. If the notification is double tapped, or pressed on, the user is redirected to the notification center.

Notification entries are also intractable via left swiping. Once an entry in the list of notifications is left swiped, as shown in figure 14 the user can perform two actions. They, and their functionalities, are as follows:

1. **Trash/Delete**: Removes the notification and presents the user with the option of blocking all future notifications for that specific permission.
2. **Settings**: Much like the **"GO TO SETTINGS"** button on the previously discussed permissions page popup dialog, this button redirects the user to the Android permission management page for the related notifications application.

If the trash/delete action is selected the user is also given the option of disabling future related notifications for a specific permission. If they select to disable or block notifications, neither the notification center nor the Android OS will show DroidNet notifications until that permission is removed from the block list.

Additionally, identical to the permissions page, if a user clicks the settings button they are brought to the permissions associated Android settings page. Then, as aforementioned, a popup dialog reminding the user of what permission requires changing and DroidNet's confidence for the recommendation is displayed as an overlay (see figure 15).

3.3 Server-side Implementation

The DroidNet network and architecture requires various operations to be performed by the client. Specifically, DroidNet utilizes a database and back-end algorithm for permission recommendation as well as user rating. This section, and the following subsections, cover those relationships and explain their implementation and importance.

3.3.1 Database

A database is needed to store entries submitted from users upon completion of the synchronization process discussed in subsection 3.2.4. Three tables exist in DroidNet's primary database: (1) the main table, responsible for storing sensitive user information, such as their email, age, and hashed device identifier; (2) the permissions table, which contains a list of every users' permissions and whether they were granted or denied; (3) a generated list of permission recommendations for every app able to receive them.

These tables are populated in two different ways. Method one is through POST queries using Android Volley, which is a HyperText Transfer Protocol (HTTP) library. The alternative method is through string requests done through queues. Alternatively, requests can be sent and values inserted through the usage of the Android

HTTPURLConnection class.

The requests are then sent through a placeholder website using cPanel for hosting and GoDaddy.com for domain registry. The SQL database can be accessed through a phpMyAdmin portal. The key data collected from each query contains information relating to the user and permission data and matches the pattern <*ID, App Name, Permission, Value*>. The tuple can be seen in figure 7, at the beginning of this chapter, which describes DroidNet's architectural design.

3.3.2 Algorithm Implementation

The algorithm used for producing recommendations consists of the following:

1. Create "User" and "Application" objects by reading in CSV files from the database and storing their values in each object respectively.
2. Select C_0 value.
3. Rate direct users by comparing them to the permissions of any existing seed expert(s) described in subsection 2.4.5.
4. Rate the remaining users by treating them as indirect neighbors and using the iterative method described in subsection 2.4.5.
5. Label any remaining users as un-rateable after the users in round t-1 and t remain the same.
6. Select τ_e and τ_e threshold values.
7. Display recommendations for all or selected applications.

3.3.3 User Rating Algorithm Modifications

The original DroidNet paper developed a rating algorithm for a generated dataset and failed to take into account user behavior relating to Android apps. Specifically, permissions requests are made individually rather than all being initially shown at once. As aforementioned, Android apps come with their permissions initially denied. Users must make decisions for permissions (see subsection 2.2.1) as events related to those permissions occur in the application environment.

The original assumption looked at users and made comparisons between every application's permission and DroidNet's seed experts permissions. However, it may be the case that a user never encountered a permission through the course of using an application. Thus, if a user is not presented an option to grant or deny a permission, that permissions settings (generally denied) is not indicative of their decision, nor should impact their true expertise level.

The modified algorithm only compares permissions that users have granted, or approved, to those of seed experts. This modification allows filtration for apps and permissions that were installed by users, but never accessed or used. While this results in fewer overall comparisons, it in turn reflects in a more accurate representation of a user's decisions shown through their final rating.

Additionally, this works implementation of the DroidNet recommendation algorithm puts zero emphasis on a user's generated confidence level. Users are also not require to have a minimum amount of installed apps or permissions as a threshold to participate in permission recommendations. It is important to note that as a result, there is no requirement other than a single recommendation being made by one or more users above the expertise threshold for DroidNet to generate a recommendation for any given permission.

CHAPTER 4

EVALUATION

This section contains the bulk of the paper's novel contribution. Additionally, it includes statistical information relevant to the collected dataset, such as the amount of users, permissions, age groups, and other relevant points. The following subsections discuss methodologies and results sfor the core evaluation done on the implementation of DroidNet in order to verify its implementation and design.

4.1 Data Collection

Data for evaluation purposes was collected through the official DroidNet application advertised through Amazon Mechanical Turk and the Google Play store. Users who took the survey were instructed to simply install the application, wait a short amount of time for a verification code and the synchronization process, responsible for inserting data, to be complete. Once they verified they had downloaded the application through inputing a generated verification code, unique to each participant, into a text field available through survey, they were instructed to uninstall the application from their device(s). If the verification code they reported matched a column that was inserted into one of DroidNet's online database tables they received a monetary reward as compensation. Participants had full notice of what information was being collected, specifically that no sensitive or identifiable information would be stored.

Once the DroidNet application is open, logged in, and registered to a device and user, the application will attempt to synchronize with an online database to query and insert entries. The synchronization process, which is completed on application

Total GT Apps	55
Total Unique Users	307
Total Unique Apps	5599
Total Unique Permissions	10
Total Unique App-Permission Combinations	18369
Total Permissions Collected	58385
Average Number of Permissions Requested Per App	3

Table 3.: User and Permission Dataset Statistics

load up, can be seen in figure 16. These values are inserted into the database in one of the two methods mentioned in subsection 3.3.1.

4.1.1 Statistical Relevance (Dataset Statistics)

Before analysis, data integrity and diversity must first be examined to assure that enough rating rounds were completed, and a sufficient spread of user types present for an accurate evaluation when examining recommendations.

As shown in Table 3, on average each application requests three unique permissions. It also shows the total numbers of individual users in the dataset to be 307, which is sufficiently large enough that the true rating distribution of the network, or at least an apparent curve, should be visible after analysis. Additionally, it also shows relevant datapoints relating to collected permissions and application installs.

Demographic data, specifically age, is shown in pie chart form (see figure 17). Gender, region information, education level, or other demographics were not recorded. Initially, this type of data was collected under the assumption that region based statistics would be present through the console for managing and viewing user statistics. However, after further examination, sufficient region statistics are not present and will not be looked at in the remainder of this paper.

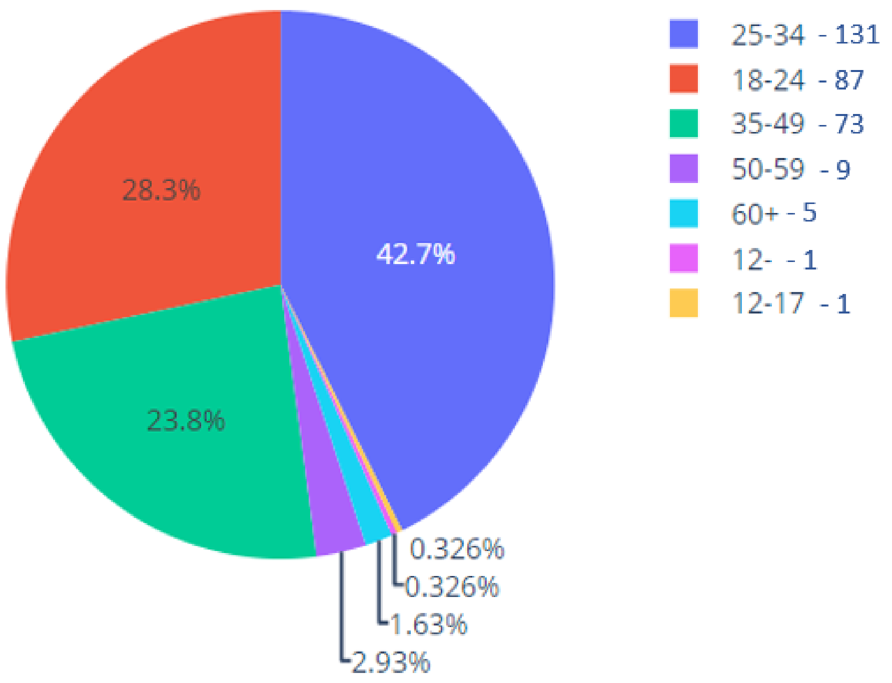


Fig. 17.: Age Group Distribution

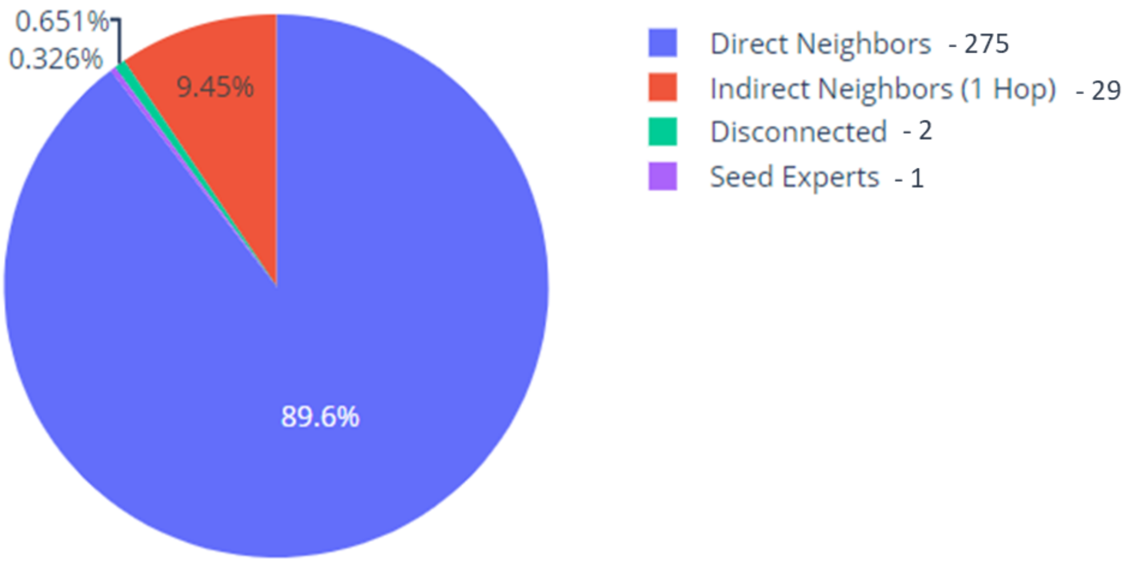


Fig. 18.: User Type Distribution

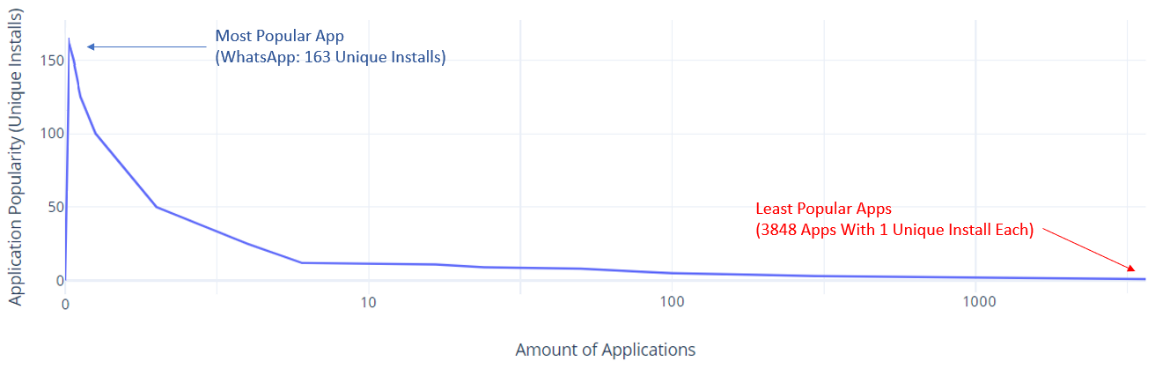


Fig. 19.: Application Install Popularity

Additionally, the total amount of users and their distribution throughout the network is shown as a pie chart in figure 18. As aforementioned, this information is important in analyzing how different user types impact not only the overall network average, but also how indirect and direct user's ratings influence each other. For further topology based definitions see section 2.4.

Figure 19 and 21 display the amount of apps downloaded by individual users in terms of popularity. In figure 19, which has had its x-axis logarithmically scaled for clarity, the left side of the graph shows the frequency of unique apps with the most installs, where the right side shows the frequency of the least installed apps. It is worth noting that 3848/5599 apps were unique to their respective users (see figure 19).

Figure 20 shows the distribution of the permission groups in the collected dataset. Storage access is by far the most requested permission group, appearing 13128 times out of 58385 application permissions, with a 22.49% request rate. Body sensors appeared to be the least requested group, being requested 157 times, or 0.269% of the time.

Next, the impact of the initial belief value on the distribution of the network generated through the mentioned dataset will be looked at.

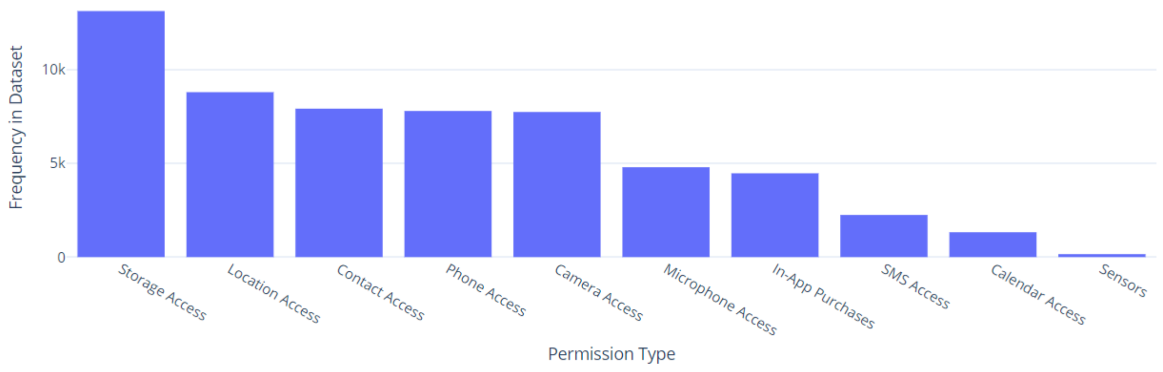


Fig. 20.: Frequency of Permissions in Dataset

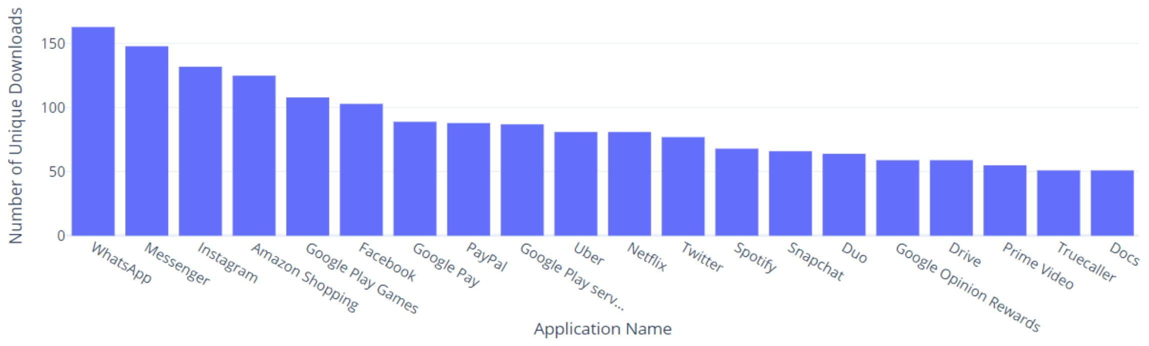


Fig. 21.: 20 Most Popular Installed Applications from Dataset

4.2 Impact of Initial Belief Values

The initial belief, C_0, is a constant value used for normalization and artificial rating reduction for users, as described in 2.4.5. Other factors that impact rating are required to be kept consistent to properly observe the impact of the initial belief on the rating distribution. As the rating threshold largely impacts the distribution of user ratings throughout the network, a constant value must be selected for it. The original DroidNet paper selects the RT value to be kept constant at $\frac{1}{2}$. Thus, the proceeding sections will do so as well in an attempt to model the behavior shown in the original work.

Type \ Constant	2	5	10
Relaxed	✓		
Normal		✓	
Aggressive			✓

Table 4.: Values for Initial Belief C_0

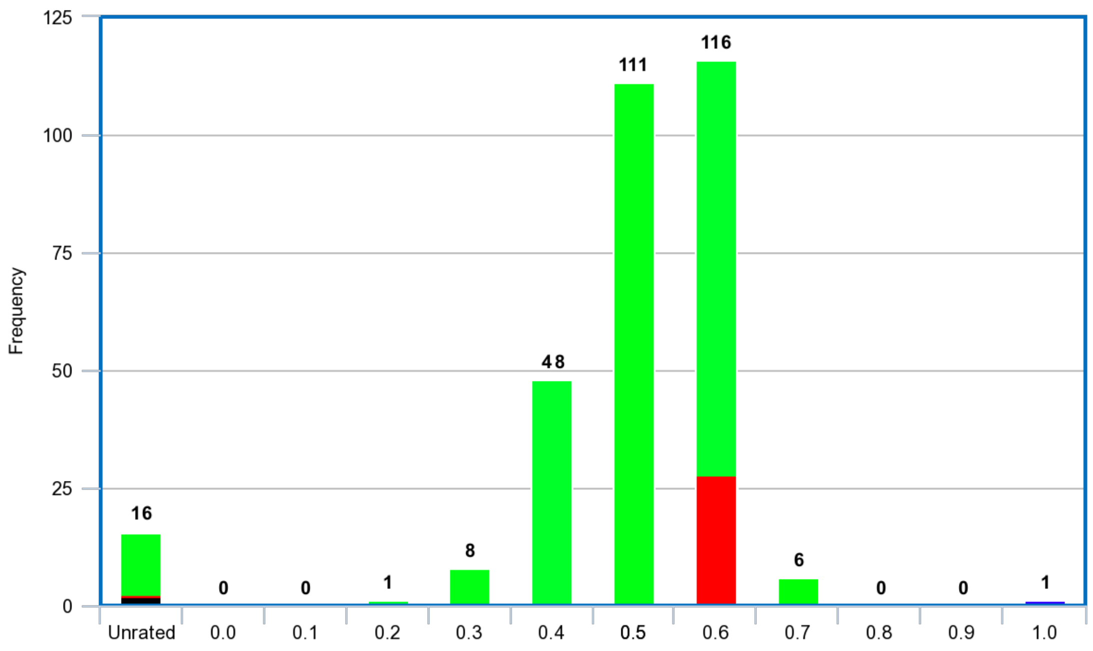


Fig. 22.: Calculated User Expertise Distribution for RT = 0.5 and $C_0 = 5$

Table 4 shows the values of the initial belief that this subsection will examine. As the initial belief impacts the networks rating distribution, each distribution generated from the C_0 values shown in the table will be analyzed and compared. Then, comparisons between all the distributions will be made and the most appropriate value selected to remain constant for the remainder of this papers evaluation.

First, the C_0 value of five is examined, as it is the median value of the three initial belief values chosen to be analyzed. The rounded down distribution of the network

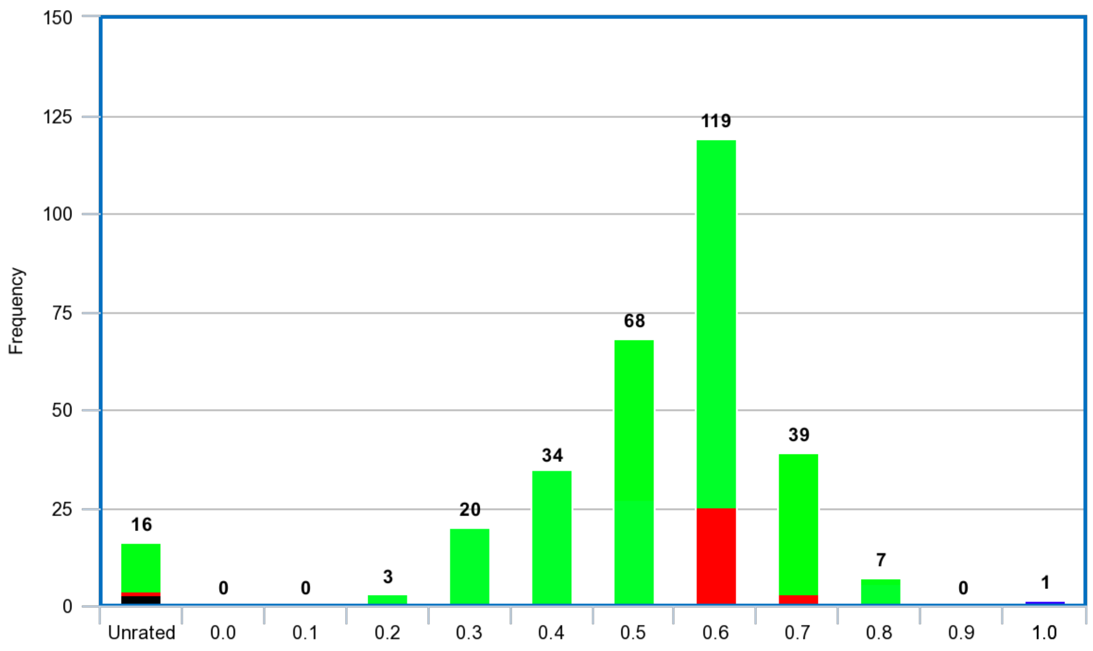


Fig. 23.: Calculated User Expertise Distribution for RT $= 0.5$ and $C_0 = 2$

ratings can be seen in figure 22. Columns labeled in red represent indirect users, black represent disconnected users, green represent direct users, and blue represent seed experts. From here, it is observed that 234 out of 307 users, or 76.22%, would be considered to have a sufficient rating by our current variable values.

Analyzing this, roughly $\frac{2}{3}$ of users are considered to have sufficient expertise by the networks standards, meaning that they are able to participate in the score propagation concerning unrated users. This increases the potential amount of apps that can later have permission recommended to them, as a larger amount of users surpassing the initial RT allows for more indirect neighbors to receive ratings, thus allowing their apps to be added into the pool of apps with rated users.

Next, the relaxed value, of the initial belief is used. Figure 23 shows a shift to the right on the network, taking on a more left-tail centric curve as opposed to the bell shaped curve seen in figure 22. However, the amount of users above the RT did

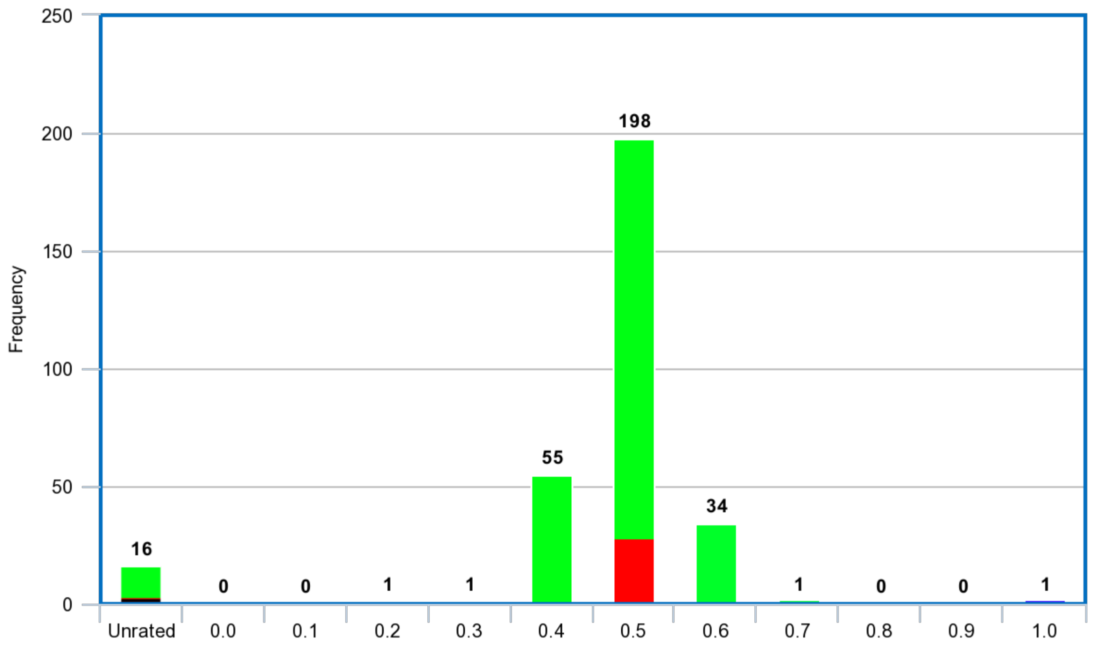


Fig. 24.: Calculated User Expertise Distribution for RT = 0.5 and $C_0 = 10$

not shift at all, regardless of the distributions slight change, remaining at 234.

To understand why, the purpose of the initial belief must be understood. The initial belief is used to add a weighted constant to the number of correct and incorrect responses a user makes when compared to another user. This is used to guarantee that the denominator in the equation for rating calculation always exceeds the numerator, even with both users matching every permission setting with one another. The relaxed C_0 value of two is inconsequential, and compared to the normal value used previously, discussing the impact of the RT on the network, the value of the initial belief was decreased. This inflates users' scores as there is a lessening of gaps between users and their direct neighbors providing them ratings.

Finally, the aggressive C_0 value is evaluated, altering the initial belief from two to 10. Figure 24 shows a large shift in the rating distribution on the network towards the center. The shape goes from being roughly bell curved to an extremely thin

Rating \ C_0	2	5	10
Average	0.5972	0.5673	0.5452

Table 5.: Average User Ratings Modifying Initial Belief

curve with users peaking in the average rating group of 0.5 to 0.6. Much like the previous C_0 value, the amount of users above the RT saw no change remaining at 234. However, the amount of users rated above 0.6 saw an extreme shift with most users being placed into the 0.5 to 0.6 range.

A key observation to note about this user shift is that while the amount of users above the RT may not have changed in any of the distributions, the average rating experiences significant change (see Table 5). It is worth mentioning that the average rating decreasing may not significantly impact the networks range distribution with the current RT of $\frac{1}{2}$, as it remains greater with every value of C_0. However, it indicates that an increased value of the initial belief translates to a normalization of the network with most users falling into the expected average rating category of 0.5. This limits the number of expert users substantially for the next subsection, where users are required to have a minimum expertise in order to participate in permission recommendation.

It becomes clear that the initial belief has a substantial impact on the networks distribution, but only in the sense that its peak increases respective to its C_0 value. Thus, the the distribution converges towards the "true" average user rating. The distribution of users that most represents a left-tailed distribution is ideal, due to the inherent nature of users with higher expertise recommending a permissions correct setting at a more consistent rate. As the relaxed C_0 value of two provides the distribution with the highest average rated user, as well as the highest rated individual users, the remaining evaluation proceeds by keeping the initial belief at a constant value of two.

4.3 Metric Analysis and Discussion of Practicality

This section focuses on comparing a pre-selected set of popular apps and their permissions with DroidNet's recommendation system. Two metrics, accuracy and coverage, are utilized in order to evaluate and discern whether the recommendations provided by DroidNet are valid or not. Both the accuracy and coverage of recommendations are dependent on multiple factors, as described in section 2.4.7.

Accuracy is defined as the quality or state of being correct or precise, and is calculated by examining the total number of correct decisions or actions over the total number of decisions possible ($\frac{MatchingPermissions}{TotalPermissions}$). To calculate accuracy, in comparison to another user or permission recommendation, simply add up the number of matching permissions and divide by the total number of permissions an app requests. The following sections use accuracy as a metric to evaluate whether or not DroidNet is capable of making relevant and appropriate decisions on various types of apps.

When calculating accuracy, DroidNet not providing a recommendation is handled by using a permission removal technique. Removal can be summarized as removing the permission receiving a NRA status from the comparison pool. Thus, said permission will not be counted as a mismatched permission. For example, if DroidNet has NRA for the "Location" permission on the "Duo" application, but a seed expert has a recommendation of "Deny" the total number of permissions being compared to will be reduced by one and the "Location" permission ignored in the final accuracy calculation. While removal may not negatively influence accuracy, it will still impact the coverage metric.

Coverage is defined as the percentage of an app's permission recommendations re-

q	Remembering Factor
RT	Rating Threshold
C_0	Initial Belief
τ_e	Expertise Threshold
τ_d	Decision/Permission Recommendation Threshold

Table 6.: Variable and Threshold Definitions

ceiving a recommendation of either grant or deny by DroidNet. As aforementioned, if a permission receives no recommendation coverage will be decreased. Using the same example from above, if the "Duo" application has four permissions and one receives no recommendation while the other three receive recommendations by DroidNet the total coverage will be $\frac{3}{4}$, or 75%.

In order to verify the quality of evaluation regarding permissions, the following two variables are altered in the following subsection to show their individual impact on the network:

1. The expertise threshold τ_e.
2. The decision threshold, τ_d, used for permission recommendations.

Two different types of analysis in regards to τ_d and τ_e are done. First, a broader range of threshold values for τ_d and τ_e are looked at to view the impact of both coverage and accuracy regarding recommendations. Combinations of these variables can be seen in Table 7. Following this, the most optimal τ_d and τ_e values for both accuracy and coverage will be selected after viewing their impact on both metrics. Again, these two sets of values will be kept constant, in addition to the other aforementioned values, while examining a small subset of apps.

Table 6 reviews the definitions for all included variables and thresholds of importance. As a note, a τ_d of 0.99 is represented as 1.0 for consistency.

τ_e	τ_d
0.0	0.5
0.1	0.6
0.2	0.7
0.3	0.8
0.4	0.9
0.5	1.0 (*0.99)
0.6	
0.7	
0.8	
0.825	
0.9	
1.0	

Table 7.: τ_e and τ_d Combinations

4.3.1 Coverage of Permission Recommendations

In order to assess the coverage of recommendations made by DroidNet, multiple subsets of GT apps, or apps that received recommendations by the seed expert(s), were examined. The subsets contained apps from the initial set of GT apps, but were selected at random over 50 individual trials. The selection of these apps directly impacts the rating distribution on the network, as apps included in the GT determine direct users, who in turn determine and rate indirect users. In total, 5559 unique apps and their possible 18369 total permissions were analyzed (meaning a unique total of 18369 app-permission combinations).

Figure 25 shows the coverage of the network dependent on the amount of GT apps initially selected. The first subset consists of a single random app from the set of GT apps. The following subset contains 11 with each subsequent subset containing 11 more apps. The final set examines all 55 GT apps provided by the seed expert(s). As the amount of apps included in the GT inherently impacts the rating distribution

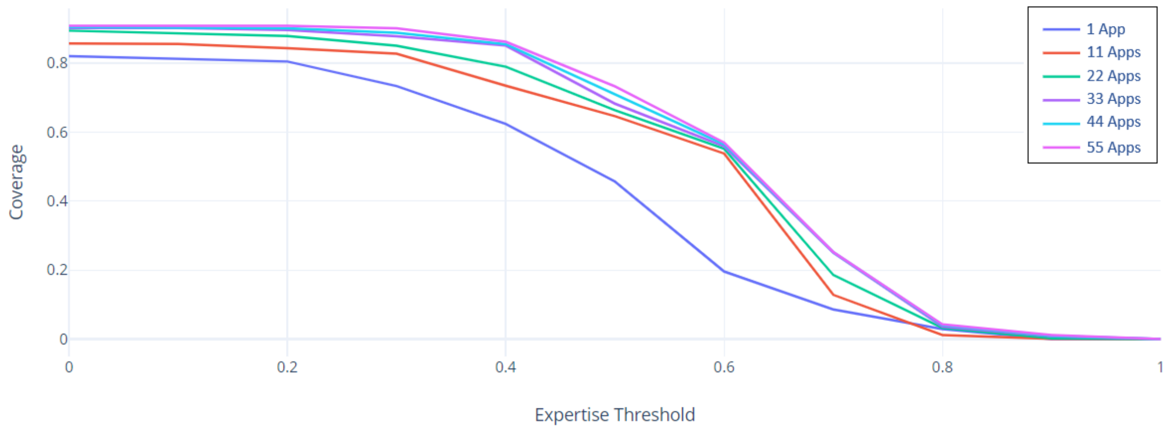


Fig. 25.: Full Permission Coverage Dependent on Amount of GT Apps with $\tau_d = 0.5$

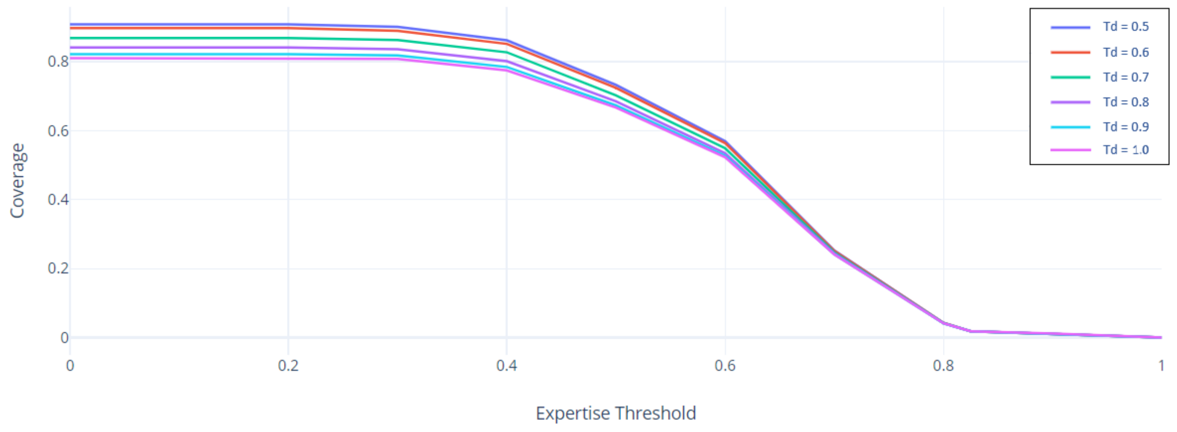


Fig. 26.: Full Permission Coverage Using All GT Apps (55) with Varying τ_d

on the network, factors such as the C_0 and τ_d were kept constant. The C_0 was kept constant at two, as aforementioned, and τ_d at 0.5.

Upon initial glance, from figure 25 it appears that the amount of apps selected to be included in the set of GT apps have only a slight impact on the coverage of permissions provided by DroidNet's recommendations. However, it can be clearly seen that as the number of GT apps increases, the coverage of permissions recommended by DroidNet increases respectively. Multiple explanations for this slight, rather than drastic, impact can be extrapolated.

One explanation is the interconnectedness of the network. This indicates that most users are able to be rated indirectly regardless of the amount of direct users. Additionally, coverage already starts relatively high. To understand the true impact of the amount of GT apps, the percentage increase from the original curve of each subset of apps should be examined rather than looking at the overall coverage percent. Coverage begins to converge as the number of apps included in the GT approaches the maximum size of the original set for each τ_e value.

Another contributing factor is that the τ_d value was kept at the bare minimum, effectively removing all recommendations of NRA, resulting in full coverage for an application and its permissions occurring when even one user provided a permissions input. As a result, it can be concluded that an app's permission will be included in the coverage of DroidNet recommendations regardless of the settings provided by users for a τ_d of 0.5.

Figure 26 shows the extended coverage of all GT apps, as also displayed by the last line in figure 25, modifying the τ_d. As seen in figure 26, coverage initially starts high as most individuals can participate in recommendations due to a low τ_e threshold, meaning more permissions receive recommendations. However, as both the τ_d and τ_e increase the coverage of recommendations made by DroidNet decreases. This can be explained by examining τ_d and τ_e individually.

Both figure 25 and figure 26 indicate that as τ_e increases there may be no user of sufficient rating other than the seed expert(s) available to participate in providing recommendations. This would lead to only the small subset of GT apps receiving recommendations, and these recommendations would be the permission settings of the seed expert(s) themselves. As seed expert recommendations should be followed in most, if not all cases, an extremely high τ_e defeats the purpose of involving other expert users in the recommendation process.

It can be observed in figure 26 that coverage follows the same curve for nearly every value of τ_d. An explanation for this would be the fact that the majority of apps on the network were downloaded by one or a few users, meaning that said user(s) would be one of few users participating in that app's permissions recommendations. This greatly increases the chance for agreement for what settings a permission should follow, as less users participating in rating increases the likeliness of full or majority agreement. As stated prior, as τ_d increases the total range of confidence values in which a permission may receive a recommendation of "Grant" or "Deny" decreases, inverse to "NRA". Notably, a τ_d of 1.0 identifies permissions that receive 100% agreement between users.

Additionally, we must remember that the GT set only contains permission settings for 1% of total apps, and while it may play an important role on initial user ratings, its impact is largely dependent on the inclusion of the more popular apps in its set. It should also be noted that 3848 apps were only installed by one user each (see subsection 4.1.1 figure 19). As long as said user and their expertise level exceed τ_e that application will effectively considered covered.

Thus, it can be concluded that the amount of apps included in the initial GT set has a minor impact on the coverage of DroidNet's recommendations, but it should be noted that values converge as the amount of apps included in the GT set increase. Additionally, both the τ_e and τ_d have noticeable impact on the coverage of DroidNet's recommendations. As values for both thresholds increase coverage decreases respectively. Next, the impact of accuracy on the set of ground truth permissions will be looked at.

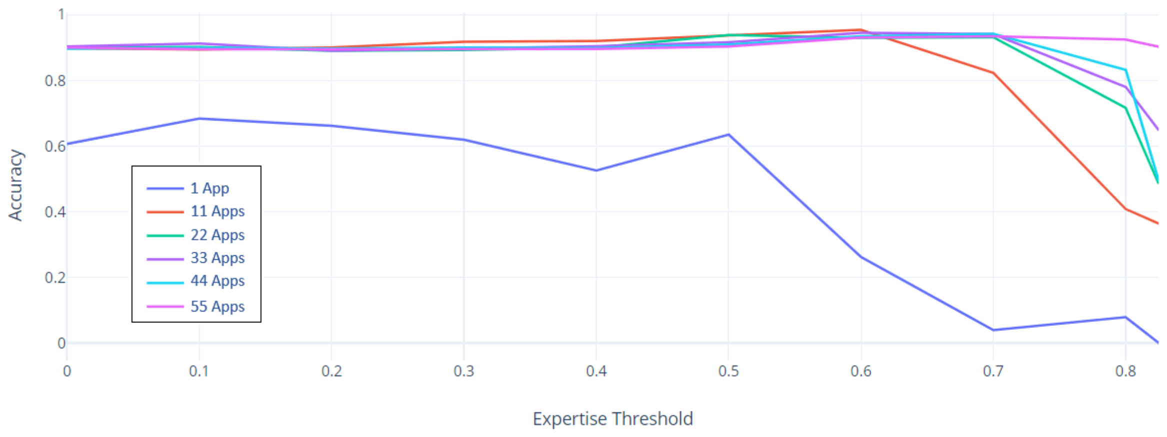


Fig. 27.: Accuracy Coverage Dependent on Amount of GT Apps with $\tau_d = 0.7$

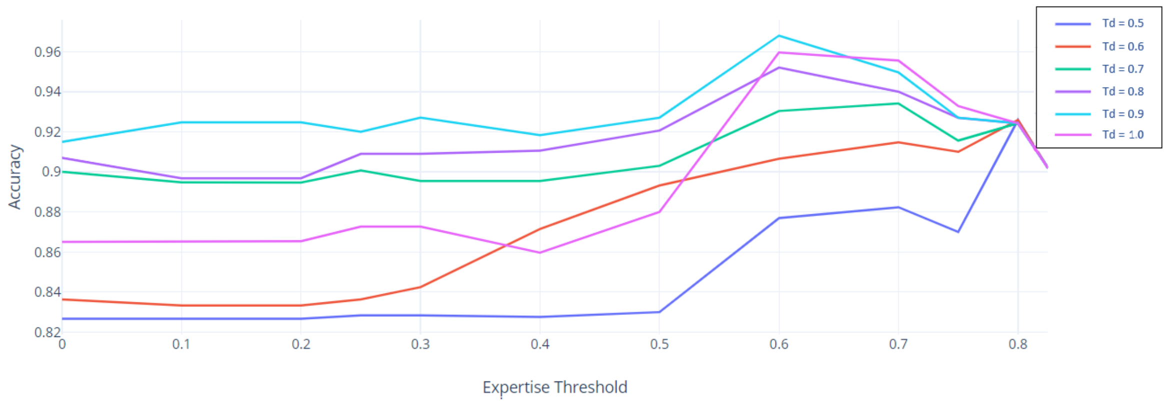


Fig. 28.: Accuracy for Complete Set of GT Apps

4.3.2 Accuracy of Recommendations for Ground Truth

For accuracy calculations, the 55 unique apps included in the original GT and their 216 respective permissions were analyzed and compared to DroidNet's recommendations. It is important to note that during calculations, seed experts were removed from the pool of participants used for recommendation as they would falsely inflate accuracy. Similar to the coverage based trials, multiple subsets of the 55 ground truth applications were tested over 50 individual trials.

Figure 27 shows the accuracy of the network dependent on the amount of GT apps initially selected. The first subset consists of one random app from the set of GT apps. The next contains 11 apps, with each subsequent subset containing 11 more. The final set examines all 55 GT apps provided by the seed expert(s). It is clear that only a small subset of GT apps are required for the accuracy of the network to converge independent of the expertise threshold. The C_0 was kept constant at two, as aforementioned, and τ_d at 0.7 (different from the 0.5 used for coverage calculations).

Figure 28 shows the accuracy of recommendations in terms of comparisons made to the complete set of GT apps. The apparent curve of the graph, and each respective τ_d line, indicates that as both the τ_d and τ_e increase the accuracy of recommendations made by DroidNet increases as well. This is due to restricting the number of expert users by threshold τ_e, as expert users are generally more likely to make fewer incorrect decisions. However, this can also be due to the inherent number of users close to the maximum expertise rating. As τ_e approaches 1.0 the only user(s) on the network capable of making decisions are the initial seed experts. As seed experts are always assumed to be correct, the confidence for their recommendations will as well be high making τ_d effectively irrelevant.

Interestingly, it can be observed that accuracy decreases if τ_e is selected too high. This can be attributed to the user rating distribution on the network. As the networks distribution was selected to be left-tailed due to the selected C_0 value (see subsection 4.2), most users fall into the 0.6-0.8 rating range. As the expertise threshold increases, fewer users are able to participate in permission recommendations and the accuracy of the network begins to approach the true rating of a few selected expert users. This finding can be isolated and observed by selecting a τ_e value of 0.825. This τ_e value results in only one user being used for recommendations, as the highest rated user on the network has a rating of 0.827.

RT	C_0	q
0.5	2	1

Table 8.: Variable Values Kept Constant for Permission Recommendation(s)

4.3.3 DroidNet Single Case Recommendation Comparisons

The remainder of this section analyzes actual DroidNet recommendations based on the collected dataset mentioned in subsection 4.1.1. Specifically, it examines the importance of the expertise and decision thresholds on the quality of recommendations provided by DroidNet using the aforementioned metrics of accuracy and coverage. Both the highest seen individual τ_e and τ_d combinations for accuracy and coverage, selected respectively from figures 28 and 26, are chosen to be analyzed.

Relatively optimal constant values are used for the variables discussed in the previous sections. A value of two, as discovered from section 4.2, for the C_0 is used. The remembering factor q, for weighting influence of permissions based on experience, is kept constant at one. A value of one means all permissions are weighted the same, regardless of how recent an observation may be. These constant values can be seen in Table 8, and the formula for permission recommendation can be reviewed in subsection 2.4.7.

The set of apps used in this subsection were included in DroidNet's initial set of GT apps provided by seed experts, meaning these apps had a preselected set of recommendations for each of their permissions that are assumed to be correct. It is important to note that while these initial recommendations held a higher weight in the initial selection and rating process for each user on the network, they do not serve as a guaranteed prediction for the output of DroidNet's recommendation system. Aggregate recommendations by normal, expert, users can outweigh even seed experts,

Ground Truth Applications and Selected Permissions			
Shazam	Tinder	Snapchat	Messenger
Camera - Deny Location - Deny Microphone - Grant Storage - Deny	In-App Purchases - Grant Location - Grant Storage - Grant	Camera - Grant Contacts - Grant Location - Deny Microphone - Deny Phone Access - Deny Storage - Grant In-App Purchases - Grant	Phone Access - Deny SMS Access - Deny In-App Purchases - Grant Storage - Grant Location - Grant Calendar Access - Deny Contacts - Grant Camera - Grant Microphone - Grant

Table 9.: Seed Expert Permission Decisions

but are rare.

Table 9 shows four different apps and their respective permissions included in the set of GT apps. These apps were selected due to their functionality and permissions. Each individual application utilizes one or more specific, identifiable permissions and cannot function properly without them being granted. Additionally, each application was downloaded by a significant amount of unique users on the network (generally over 10%). This precaution was taken in order to avoid inflated recommendation scores as a potential result of only a few users participating in the recommendation process. The following is a brief description of what each apps purpose is, as to explain why specific permission functionality relating to permissions is required:

- Shazam: A music based application that requires the use of a devices microphone in order to identify music/songs.
- Tinder: A dating application that requires the use of a devices current location in order to identify other users in a selected nearby radius.

- Snapchat: A photo and video sharing social media application that requires the use of a devices camera in order to send pictures.
- Messenger: A instant messaging application developed by social media giant Facebook that allows its members to interact with other users on its network.

In the following subsections recommendations of NRA are highlighted in orange, while mismatched recommendations are highlighted in red. Matching recommendations remain uncolored in black text.

Deny	NRA	Grant
(0.0 - 0.5)	N/A	(0.501 - 1.0)

Table 10.: Permission Recommendation Ranges with $\tau_d = 0.5$

4.3.3.1 Coverage Based Experimental Results

Table 10 shows the ranges for each recommendation type, which can be viewed in each of the following tables next to each app's respective permissions. Most importantly, Table 11 shows all of the coverage based recommendations generated by DroidNet.

Here, an expertise threshold of 0.0 and permission recommendation threshold of 0.5 are used to allow participation from every user with a given application installed. After calculations, Table 13 shows the accuracy of recommendations provided by DroidNet to be $\frac{19}{23}$, meaning a 82.61% permission match rate when compared to Table 9. Coverage was calculated to be $\frac{23}{23}$, or 100% coverage of permissions (see Table 13).

While every permission was covered by DroidNet, it can be seen by the permissions highlighted in red that a significant amount were recommended incorrect or mismatched settings. Rather than provide more recommendations, users should be provided with a greater amount of correct recommendations. From this we can conclude that drastically increased, or more specifically complete, coverage allows DroidNet the opportunity to potentially recommend incorrect settings due to lack of accountability when involving lower rated users.

Selected Application and Permission Recommendations for Highest Coverage			
Shazam	Tinder	Snapchat	Messenger
Camera - Deny	In-App Purchases - Grant	Camera - Grant	Phone Access - Deny
Location - Deny	Location - Grant	Contacts - Grant	SMS Access - Deny
Microphone - Grant	Storage - Grant	Location - Grant	In-App Purchases - Grant
Storage - Deny		Microphone - Grant	Storage - Grant
		Phone Access - Grant	Location - Deny
		Storage - Grant	Calendar Access - Deny
		In-App Purchases - Grant	Contacts - Deny
			Camera - Grant
			Microphone - Grant

Table 11.: Results for Coverage Oriented τ_e and τ_d

Status / App	Match	Mismatch	Percent Match
Shazam	4/4	0/4	100%
Tinder	3/3	0/3	100%
Snapchat	4/7	3/7	57.86%
Messenger	8/9	1/9	88.88%

Table 12.: Accuracy for $\tau_e = 0.0$ and $\tau_d = 0.5$

Status / App	Covered	Not Covered	Percent Covered
Shazam	4/4	0/4	100%
Tinder	3/3	0/3	100%
Snapchat	7/7	0/7	100%
Messenger	9/9	0/9	100%

Table 13.: Coverage for $\tau_e = 0.0$ and $\tau_d = 0.5$

Deny	NRA	Grant
(0.0 - 0.1)	(0.101 - 0.9)	(0.901 - 1.0)

Table 14.: Permission Recommendation Ranges with $\tau_d = 0.9$

4.3.3.2 Accuracy Based Experimental Results

Similar to the previous subsection concerning coverage, Table 14 shows the ranges for each recommendation type. Again, the recommendation results can be viewed in Table 15.

The accuracy of recommendations provided by DroidNet for the four apps in question when using an expertise threshold of 0.6 and permission recommendation threshold of 0.9 can be seen in Table 16. After calculations, accuracy results in a score of $\frac{12}{12}$, meaning a 100% permission match rate. Coverage, as shown in Table 17, was calculated to be $\frac{12}{23}$, or 52.17% coverage of all selected permissions.

From this we can see DroidNet provides perfect accuracy when recommendations are compared to the GT. However, coverage drops significantly to nearly half of the available permissions. It can be concluded that the increased accuracy is due to the involvement of an expertise threshold greater than 0.0. The trade off in coverage for increased can largely be considered worth it, as false permission recommendations pose greater harm than a lack of recommendations or incorrect ones.

Thus, it can be concluded that coupled with the coverage based recommendations, DroidNet is able to provide correct recommendations for a majority of apps given the right threshold values for both τ_e and τ_d.

Selected Application and Permission Recommendations for Highest Accuracy			
Shazam	Tinder	Snapchat	Messenger
Camera - Deny	In-App Purchases - Grant	Camera - Grant	Phone Access - Deny
Location - NRA	Location - Grant	Contacts - NRA	SMS Access - Deny
Microphone - Grant	Storage - NRA	Location - NRA	In-App Purchases - Grant
Storage - Deny		Microphone - NRA	Storage - NRA
		Phone Access - NRA	Location - NRA
		Storage - Grant	Calendar Access - Deny
		In-App Purchases - Grant	Contacts - NRA
			Camera - NRA
			Microphone - NRA

Table 15.: Results for Accuracy Oriented τ_{e} and τ_{d}

Status App	Match	Mismatch	Percent Match
Shazam	3/3	0/3	100%
Tinder	2/2	0/2	100%
Snapchat	3/3	0/3	100%
Messenger	4/4	0/4	100%

Table 16.: Accuracy for $\tau_{\mathrm{e}} = 0.6$ and $\tau_{\mathrm{d}} = 0.9$

Status App	Covered	Not Covered	Percent Covered
Shazam	3/4	1/4	75.0%
Tinder	2/3	1/3	66.6%
Snapchat	3/7	4/7	42.86%
Messenger	4/9	5/9	44.44%

Table 17.: Coverage for $\tau_{\mathrm{e}} = 0.6$ and $\tau_{\mathrm{d}} = 0.9$

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

This thesis focused on both the evaluation and implementation of a proposed crowd-source based permission recommendation framework named DroidNet. The need for such a system is detailed greatly, and emphasized due to the ever growing, relatively unmonitored, mobile application market. This work also provided the collection of a unique real-world dataset containing a large amount of Android permission settings.

Additionally, the full implementation of the project and the impracticalities of the original work are described. This includes both the implementation of the software and all of its associated architectural components. Alterations to multiple algorithms were also made with reasoning as to why they were done.

In-depth evaluation of the proposed recommendation system was also completed. Two metrics, accuracy and coverage, were identified as valid indicators of high quality recommendation results and were analyzed extensively over multiple areas. Variables of great importance involved in the recommendation process were also analyzed in order to identify values and trends of significance.

The results of this work show that it is more than feasible to create a permission recommendation system while using input from both expert and regular users. This work also shows that through the initial selection of expert users other high rated users can be identified. In addition to this, it was also shown that both experienced and inexperienced users can be assigned ratings/weights that result in the generation of high quality, accurate, recommendations.

Future work and evaluation can be done in order to fine tune DroidNet's settings.

While multiple threshold and constant values were examined, a substantial amount of combinations, including the most optimal settings, can still be found. A combination of such values could potentially result in DroidNet successfully reaching both perfect coverage as well as accuracy. The collection of a larger, more diverse, dataset in addition to the examination of a greater amount of users could provide further insight into the true network rating distribution after utilizing DroidNet's user rating algorithm.

Additionally, changes to initial seed expert selection, and consequently the applications covered by the ground truth, can be made to improve the integrity of recommendations. In the case that an unfit, or malicious, seed expert is selected for the initial user rating process, a group consensus model can be implemented to vet potential seed experts. In this model, the set of potential seed experts can compare overlapping applications in order to identify a potential expert with incorrect recommendations. If identified, users masquerading as experts are removed from the initial candidate selection of seed expert users.

Lastly, further work can be done concerning the recommendations provided by DroidNet to users. Specifically, the impact of recommendations can be analyzed by examining whether a user chooses to follow DroidNet's suggestions or ignore them.

Appendix A

ABBREVIATIONS

App	Application
OS	Operating System
GT	Ground Truth
API	Application Programming Interface
SE	Seed Expert
RT	Rating Threshold
NRA	No Recommendation Available

REFERENCES

[1] *Number of available applications in the Google Play Store from December 2009 to June 2019*. URL: https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/.

[2] *Research news*. URL: https://www.cam.ac.uk/research/news/what-isthe-price-of-free..

[3] Wilson Rothman. "Smart phone malware: The six worst offenders". In: *NBC-News.com* (Feb. 2011). URL: https://www.nbcnews.com/technolog/smart-phone-malware-six-worst-offenders-125248.

[4] Adrienne Porter Felt et al. "Android Permissions: User Attention, Comprehension, and Behavior". In: *Proceedings of the Eighth Symposium on Usable Privacy and Security*. SOUPS '12. Washington, D.C.: ACM, 2012, 3:1–3:14. ISBN: 978-1-4503-1532-6. DOI: 10.1145/2335356.2335360. URL: http://doi.acm.org/10.1145/2335356.2335360.

[5] Adrienne Porter Felt et al. "Android Permissions Demystified". In: *Proceedings of the 18th ACM Conference on Computer and Communications Security*. CCS '11. Chicago, Illinois, USA: ACM, 2011, pp. 627–638. ISBN: 978-1-4503-0948-6. DOI: 10.1145/2046707.2046779. URL: http://doi.acm.org/10.1145/2046707.2046779.

[6] Bahman Rashidi et al. "Android User Privacy Preserving through Crowdsourcing". In: *IEEE Transactions on Information Forensics and Security* (2017).

[7] Enrique Estellés-Arolas and Fernando González-Ladrón-De-Guevara. "Towards an Integrated Crowdsourcing Definition". In: *J. Inf. Sci.* 38.2 (Apr. 2012),

pp. 189–200. ISSN: 0165-5515. DOI: 10.1177/0165551512437638. URL: http://dx.doi.org/10.1177/0165551512437638.

[8] *Permissions overview : Android Developers*. URL: https://developer.android.com/guide/topics/permissions/overview.

[9] Adrienne Felt et al. "Android permissions: User attention, comprehension, and behavior". In: *SOUPS 2012 - Proceedings of the 8th Symposium on Usable Privacy and Security* (July 2012). DOI: 10.1145/2335356.2335360.

[10] Bahman Rashidi, Carol J. Fung, and Tam Vu. "RecDroid: a resource access permission control portal and recommendation service for smartphone users". In: *SPME@MobiCom*. 2014.

[11] Ron Amadeo. *App Ops: Android 4.3's Hidden App Permission Manager, Control Permissions For Individual Apps! [Update]*. July 2013. URL: https://www.androidpolice.com/2013/07/25/app-ops-android-4-3s-hidden-app-permission-manager-control-permissions-for-individual-apps/.

[12] Jialiu Lin et al. "Modeling Users' Mobile App Privacy Preferences: Restoring Usability in a Sea of Permission Settings". In: *Proceedings of the Tenth USENIX Conference on Usable Privacy and Security*. SOUPS '14. Menlo Park, CA: USENIX Association, 2014, pp. 199–212. ISBN: 978-1-931971-13-3. URL: http://dl.acm.org/citation.cfm?id=3235838.3235856.

[13] Qatrunnada Ismail et al. "Crowdsourced Exploration of Security Configurations". In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. CHI '15. Seoul, Republic of Korea: ACM, 2015, pp. 467–476. ISBN: 978-1-4503-3145-6. DOI: 10.1145/2702123.2702370. URL: http://doi.acm.org/10.1145/2702123.2702370.

[14] Shahriyar Amini. “Analyzing Mobile App Privacy Using Computation and Crowdsourcing”. In: 2014.

[15] Rui Liu et al. “PriWe: Recommendation for Privacy Settings of Mobile Apps Based on Crowdsourced Users’ Expectations”. In: July 2015. DOI: `10.1109/MobServ.2015.30`.

[16] Liu Yang et al. “Short paper: Enhancing users’ comprehension of android permissions”. In: *Proceedings of the ACM Conference on Computer and Communications Security* (Oct. 2012). DOI: `10.1145/2381934.2381940`.

[17] Yuvraj Agarwal and Malcolm Hall. “ProtectMyPrivacy: Detecting and mitigating privacy leaks on iOS devices using crowdsourcing”. In: June 2013, pp. 97–110. DOI: `10.1145/2462456.2464460`.

[18] Gordon Tullock. “Problems of Majority Voting”. In: *Journal of Political Economy* 67.6 (1959), pp. 571–579. DOI: `10.1086/258244`. URL: `https://doi.org/10.1086/258244`.

VITA

Pulkit Rustgi is currently pursuing his Master's degree in Computer Science at Virginia Commonwealth University in Richmond, Virginia. He received his Bachelor's of Science in Computer Science from Virginia Commonwealth University in the spring semester of 2017. His research is primarily focused on mobile application security, particularly relating to the Android operating system.