University of Colombo School of Computing

SCS4124 - Final Year Project in Computer Science

User Driven Model for Better Privacy Protection in Android Applications

Researcher:
K.J.S. Fernando
Index Number:
12000434

Supervisor:
Dr. Kasun de Zoysa
Co-supervisor:
Mr. Primal Wijesekara

(Original Title: Towards an Improved Permission Architecture for Android Applications)
09.09.2016

Preface

This document contains the Extended Project Proposal to be submitted to the University of Colombo School of Computing as a requirement of the subject SCS4124: Final Year Project in Computer Science. It includes an outline of the research question, problem statement in brief, goals and objectives of undertaking the research, methodology to be followed, the scope and delimitations of the project, project timeline, literature review, and a discussion of preliminary results.

Signatures:

K.J.S. Fernando	Dr. Kasun de Zoysa
___________________________	___________________________
Signature	*Signature*
___________________________	___________________________
Date	*Date*

Contents

1

Section One

1.1 Introduction

Android is an open source mobile operating system currently developed and maintained by Google Inc. The Android Open Source Project(AOSP) is the collective name for the Linux kernel, middleware components and applications which form the Android Operating System [1]. Analysis of global smartphone market share indicates that as at March 2016, Android with a market share of 60.99% is the leading mobile operating system with a lead of 29.23% over the next most popular operating system, iOS(market share 31.76%)[3]. The current version of Android is Android 6.0 M (or Marshmallow), with the SDK version 23.

Users of the Android platform can install "Applications" of different types from digital distribution platforms such as Google Play, Amazon Appstore, SlideME, Mobango etc. As at July 2015, Google's own online marketplace for applications, Google Play, had more then 1.6 million different applications available for download[58]. Through Google Play, developers can publish and distribute their applications to users of Android compatible smart-phones. When uploading applications, developers are required to specify which critical resources their applications will need access to in an XML file called the Android Manifest, which is typically found in the root folder of every Android application. Users are required to grant permission for this resource access either before installation(in versions older than Android 5.0) or during runtime(in Android 6.0 and newer devices).

1.2 Problem Background

Users of Android applications are expected to decide on how an application will be allowed to use data without a prior guideline to act as an indicator[26]. Users are asked to make privacy decisions before they start using the application, at which time they are not equipped with enough knowledge to do so. Up to Android version 5.0(Lollipop) permissions follow a model, where users are required to grant all the required permissions at install time. Permissions could not be selected or removed individually and choosing not to grant a permission resulted in the application download being canceled. Once installed, app permissions cannot be revoked.

In versions newer than Android 6.0(Marshmallow) users are allowed to install applications granting selective privacy options, which can later be toggled depending on individual needs. This has mitigated problems that existed in previous versions where users were not given the opportunity to revoke permissions that have already been granted, or choose to grant permission only when required etc. Permissions have been classified based on how "dangerous" they are and certain types of permissions are granted automatically, while other types have to be approved by a user at runtime. The Android framework is built on a Linux kernel, and each application is assigned a UID(Linux User ID) upon installation. Permission requests are connected to the UID or GID(Group ID), which provides process isolation. Android applications require users to approve a list of permissions that will be accessible by the app before installation. This is a shortcoming in an operating system with such a wide user base as users are not equipped with enough knowledge to make a decision, which leads to making uninformed decisions and compromising their own privacy later on since they have no guideline or benchmark available as an indication of how an application will use data once it is downloaded with the necessary permissions granted. Application ratings can be given by users on most marketplaces including Google PlayStore, but this is based on a host of factors which may or may not have taken privacy and security into consideration.

Similar situations with privacy on the internet has resulted in different user driven models, including the "web of trust", different kinds of which are used by social networks including Facebook, LinkedIn and Google+ for user validation and networking[25]. In the context of Android security, there are research applying user driven mechanisms to secure message passing, and identifying malware, but not for Application permission configuration[11].

1.3 Problem Statement

1.3.1 Privacy and Security

Privacy and security, although related are different concepts. Privacy is subjective; the user can decide on how private they want their data to be.However security is objective; it is concerned with 'guarding' something that is universally accepted as confidential, such as password, credit card details, pin number etc.

Due to uninformed privacy decisions taken by users when installing apps, both privacy and security are compromised. However security is primarily threatened through malware apps which access permissions without authority, whereas privacy is compromised through users unknowingly granting permissions for applications which then misuse these privileges. Therefore we will be concentrating on privacy violations that occur through permissions which have been granted by users, as there are many research currently focused on malicious code detection and improved security of Android applications.

Users have different needs with regard to privacy, which is why they should be allowed to make their own decisions. Therefore a one-size-fits-all regulating system which blocks each privacy infraction would not be ideal since Android does not have information to predict what each user wants beforehand. Privacy infractions are therefore more difficult to predict than security threats, since user preferences also have to be taken into account. Therefore an ideal solution would be to let the users make their own decisions, while providing enough information for them to do so. In the Android platform, users are asked to make decisions even before they start using the application, and lack information as to how and when the application is going to use these permissions, resulting in ill-informed decisions which are likely to violate their own expectations of privacy.

1.3.2 Permission Issues Before Android 6.0

Up to Android version 5 (Lollipop),users were not given an option to choose which permissions to grant upon application installation. Android v6 (Marshmallow) allows granting and revoking certain permissions upon installation, however, this model is not perfect, and even though it has been almost a year since Marshmallow was launched, it is only running on around 2.3% of Android devices. [20] The current all or nothing model forces users to either refrain from installing an application (no permissions granted) or to grant all permissions requested by an application. This can create problems since studies have shown that users tend to ignore the grant permission dialog since they have no choice except to avoid installing the app altogether. [64] The issue exists for inbuilt applications as well (most of which are classifiable as bloatware [43]), for example the Flashlight application inbuilt on devices running HTCs Android based Sense UI, requests all permissions that can be granted to an application, and since the app is inbuilt there is no way to uninstall/disable it other than rooting the device.

1.3.3 Context of Permission Requests

In the latest version of Android, Marshmallow, permissions can be toggled. However some issues still exist.

Research has shown that over 75% of permission requests rake place while the application is running in the foreground, background or as a service[64]. The context of a permission access, or specifically the time a permission is requested by an application can contribute towards finding whether the request is legitimate. For example research has further shown that the GPS lication indicator is only visible for 0.04% of all location access, whereas in reality applications use other permissions such as WifiState to determine the location if a device through SSID information or network tower location. Data collected through such requests breach users privacy and are used for purposes such as targeted marketing [53].

Research has shown that people are moved to base decisions on what they perceive as the reason for an application to access data[64]. This may negatively affect the revenue generation model of some applications, since legitimate permission requests, sometimes connected to the revenue generation model of an application, are denied by users who assume that the request malicious. This has affected applications such as AngryBirds, where users have denied the SMS permission to the app, resulting in a failure to unlock levels users have paid for with in-app purchase payments, and even Google's own Google+ application, where thumbnails and images become invisible to users who choose not to grant the Storage permission[47] [51].

In an ideal situation a user should know why an application is requesting a particular permission; as part of its core functionality, secondary functionality or as a method of revenue generation. However, this is not possible with the current model, since the level of information made available to users regarding application permissions is decided on by the developer. A discussion of other issues such as permission creep, capability leaks etc. which exist in the current model are outlined in the Literature Review chapter.

1.3.4 Problem in Summary

The current permission model used in Android applications does not include a centralized process for certification or testing before an app is released, and instead relies on the developer to act responsibly and request the least number of permissions that are necessary. However in most cases the privileges given to developers are misused causing capability leaks, permission creep and some other issues, the consequences of which cannot be reversed even after app uninstallation. The simplest way to stop the occurrence of privacy breaches caused by the permission model would be to let the user have more control over permissions to be granted. The problem in a nutshell is that at present users are asked to make uninformed decisions, that can have wide reaching consequences, which they are not equipped to answer.

1.4 Research Question

Different types of interactions take place in this domain; users interact with applications to use the functionality and applications interact with the Operating System to access data(this is where permissions are called). Further, to confirm the level of privacy provided by an app, users could interact with their peers to share the experience of whether the privacy provided by a particular application is adequate or not. Such interactions between applications and the operating system, applications and users, and user-user interaction and knowledge sharing can be used to create an environment where informed decisions can be made, focused on creating an improved permission architecture for Android applications and improving flexibility and control for users of the platform.

Existing research have explored app-user, app-sys interactions with differing results(refer literature review) but no one exploited the user-user interaction to give better privacy protection to users. As a first step, we want to study the viability of this approach the success compared to status quo.

1.5 Research Goal

Our research goal is to **propose and evaluate a *user-driven* privacy model to overcome the problems caused due to improper permission handling in android applications**.

1.6 Scope and Delimitations

The research will focus on Android smartphones with access to the Google PlayStore. Monitoring will be carried out for a selected number of applications from the PlayStore, and not from apps installed using third party websites, promotional links on social media or email, external markets such as the Amazon Appstore, F-Droid etc. The scale to determine trustworthiness of an application will be assigned simply based on factors such as frequency of uninstallation, whether uninstallation happened after a major privacy breach etc. Depending on resource constraints this rating may be equalized to a user assigned rating for an application as well. The information provided to the user will be limited and focused on the core data needed to determine whether an application can be installed without trust issues.

The research will focus on privacy violations that occur through authorized access. Therefore our scope is limited to the interaction between the app and the system. The operations of the application once it gets the resource being requested are beyond what we will be analyzing over the course of this research. We will be operating on the assumptions that trust cannot be misplaced, and that the Linux kernel on which the Android framework is built is completely secure.

1.7 Research Approach

The first step of the research was to collect data related to privacy preferences of users with regard to Android permissions. We obtained the data collected in a previous field study[64], which consisted of a set of 27 million permission requests collected from an annonymous group of 36 users over the period of one week. This dataset includes the name of the application from which the permission request was generated, whether the screen was on or off at the time of the request, the application being used in the foreground etc.

To gain a basic understanding of the permission preferences of the target group for the study we were planning to do(4th year undergraduates of UCSC), we collected data of permissions granted to 10 applications selected from the PlayStore using an ADB script that copied this data to our machine once a user's device was connected. We then asked them to complete a survey asking about their basic privacy preferences. Names of participants who were willing to take part in the user survey were also collected from this preliminary study.

Attempting to recreate the methodology followed in the previous field study to give our target users a device with a customized kernel that logged permission access was not successful as our devices consist of Micromax Canvas Android One phones, for which binaries could not be found. Therefore we revised the plan to use the previous dataset to generate a survey asking users to list their "trusted peer" and a list of questions with the objective of determining whether the peer and the user had similar privacy preferences. This study is still being conducted. Partial results are shown in section 2.3 below.

The next step of the research will depend on the results of the study currently being conducted, and will consist of decompiling the apk of the Google Playstore and editing the code to allow users to adopt privacy settings of a trusted peer when in doubt.

1.8 Project Timeline

Task	**Expected Date of Completion**
Preliminary meetings with the supervisor to discuss the research	Completed
Confirm supervisor details	Completed
Confirm tentative topic	Completed
Preliminary literature review	Completed
Submit Project Proposal	Completed
Proposal Defense	Completed
Submit introductory chapter of thesis	Completed
Preliminary user study and survey	Completed
Submit outline of chapter two-literature review	Completed
Study of existing dataset and building modified kernel	Completed
Submit activity plan for semester two	Completed
Survey to generate the a network of users with trusted peers	In progress
Submit extended proposal	09/09/2016
Interim defense	25/09/2016
Modify PlayStore apk	30/09/2016
Detailed outline of thesis	06/11/2016
Evaluation of results from survey and modified Playstore	15/11/2016
Submission of draft thesis	27/11/2016
Submission of final thesis	25/12/2016
Final defense	29/01/2016

Table 1.1: Tentative timeline

2

Section Two

2.1 Literature Review

2.1.1 Android Permission Model

Comparison of Application Approval Process

The leading mobile operating systems apart from Android are iOS by Apple, Windows Mobile and BlackBerry OS. Each of these have different processes and methodologies for developers to follow before submitting an application to the store. A comparison between application submission processes for these platforms shows that each has a centralized process for validation and/or verifying an application before it is made available for users to download. However these processes are usually in place for checking security or applications, and not privacy related issues.

To submit an application to iTunes, the marketplace for iOS applications, the developer is first required to create an App ID and a Distribution Provisioning Profile and then submit an application through iTunes connect with detailed information on the app.Three different certificates have to be submitted along with the application; the Distribution Certificate, Push Notification Certificate and Mobile Provisioning Certificate. The app has to be submitted through the Publication Center, where a checklist of complying standards that need to be adhered to has to be filled in.The approval process on average takes six days to one week.[36]

Windows store applications also go through a centralized process before being released. A submission has to be created for each application with a checklist of information. Once the submission is complete and the application has been preprocessed without errors, it is submitted for certification through the Windows Certification Kit. The certification process focuses on three core areas; security tests, technical compliance tests and content compliance tests. The amount of time taken for an application to receive approval fluctuates based factors such as the code and logic complexity, visual content, rating of the developer, other applications in the queue etc. Applications which fail the certification process will be returned with a report indicating where the compliance standards were not met. Developers are allowed to resubmit applications following the same process.[16]

RIM(Research In Motion) BlackBerry requires developers to submit applications for a complex process of reviewing, testing and ”readying for publication” before being awarded a Approved/Up For Sale rating. Apps with this rating can be submitted and released on the BlackBerry marketplace; BlackBerry App World.[40]

Android developers add applications to marketplaces including Google PlayStore, Amazon AppStore, GetJar, SlideMe and F-Droid, which can then be downloaded by users. The problem lies in there being no centralized security measurement for applications on such marketplaces. Developers are trusted to prepare an application for release and then release it through a marketplace, email or website.[21] The Android operating system imposes some security and privacy restrictions, including an install-time permission system, where each application declares what permissions it requires upon installation.[2] This can provide users with control over their privacy since the choice to cancel installation lies with the user.

Application Security and Privacy

The Android OS is built on top of the Linux kernel, which manages the hardware including drivers and power management. From the bottom-up, the next layers include HAL or the Hardware Abstraction Layer, Native libraries and the runtime, which includes the Dalvik virtual machine, the framework with activity managers, system view and content providers, and a layer of applications(Refer Figure 2.1). Each time a new application is installed, the operating system assigns it a unique Linux user ID, which will be persistent for the lifetime of the application on that device. On uninstallation of an application, the user ID will be freed and possibly reassigned to another new application after the device is rebooted for the first time following an uninstallation.

Permissions are tied to the user IDs, attemting to provide application sandboxing and process isolation by limiting communication between applications to methods overseen by the operating system such as passing of intents. Applications get a dedicated part of the file system connected to its UID whiich can be used to write private data and store databases and raw files, including permission information. According to the description on the Android Security website, the OS ”seeks to be the most secure and usable operating system for mobile platforms by re-purposing traditional operating system security controls to protect user data, protext sstem resources and provide application isolation by providing robust security at the OS level through the Linux kernel, mandatory application sandbox for all applications, secure interprocess communication, application signing and application-defined and user-granted permissions”[7]. However, this system has several loopholes, and malicious applications can access sensitive resources even in cases where sandboxing is done by the user ID system[45].

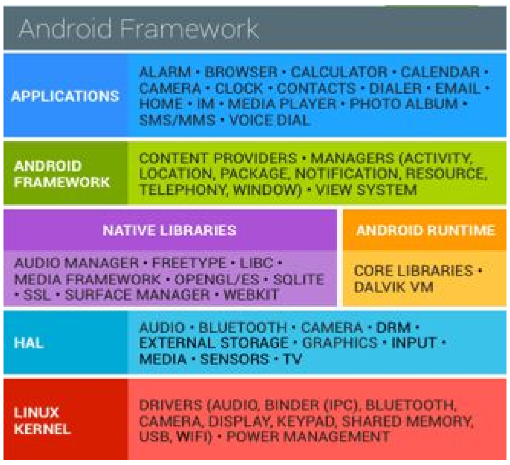

Android Framework
APPLICATIONS
ALARM • BROWSER • CALCULATOR • CALENDAR • CAMERA • CLOCK • CONTACTS • DIALER • EMAIL • HOME • IM • MEDIA PLAYER • PHOTO ALBUM • SMS/MMS • VOICE DIAL
ANDROID FRAMEWORK
CONTENT PROVIDERS • MANAGERS (ACTIVITY, LOCATION, PACKAGE, NOTIFICATION, RESOURCE, TELEPHONY, WINDOW) • VIEW SYSTEM
NATIVE LIBRARIES
AUDIO MANAGER • FREETYPE • LIBC • MEDIA FRAMEWORK • OPENGL/ES • SQLITE • SSL • SURFACE MANAGER • WEBKIT
ANDROID RUNTIME
CORE LIBRARIES • DALVIK VM
HAL
AUDIO • BLUETOOTH • CAMERA • DRM • EXTERNAL STORAGE • GRAPHICS • INPUT • MEDIA • SENSORS • TV
LINUX KERNEL
DRIVERS (AUDIO, BINDER (IPC), BLUETOOTH, CAMERA, DISPLAY, KEYPAD, SHARED MEMORY, USB, WIFI) • POWER MANAGEMENT

Figure 2.1: The Android Software Stack

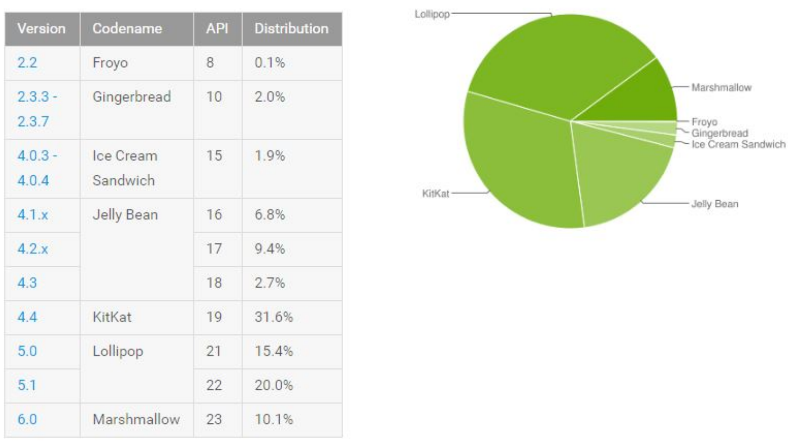
Installing Applications

The Android OS allows users to install both free and paid third party Java applications through markets such as Google Play. As discussed Android applications do not go through a review process to measure compliance to guidelines, although most competitors including iOS, Windows and RIM do. Access to sensitive resources, also known as ”permissions” are controlled by the operating system to let aplications request access to sysytem functionality through an XML manifest, forming a computer-supported-access-system(CSAS)[59]. Applications are allowed to define their own extra permissions in addition to the 134 core permissions provided by Android[1]. (This research will not focus on third party defined permissions.)

Google Play application listings show information about each application including screenshots, marketing jargon, compatibility with user devices, app information, developer information, content rating, reviews and other related applications. However this does not include a list of permissions used by the app, or any privacy or security related information on the application page. Once a user clicks on the install button, they are expected to approve or disapprove the permission list that appears when downloading on devices older than Android M, without any background information. Permissions and privacy information are not highlighted on both the mobile application of Google Play or the store website[39].

Before Android Marshmallow

Before Android sdk 23, the permission model was based on a ”do-or-die” concept. Users were given a list of permissions that would be required upon clicking the ”install” link on an application on Google Play. There were no options to grant permissions selectively or revoke when not needed, and choosing not to grant a particular permission resulted in the application installation process being terminated[27]. User choice was limited to



Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	2.0%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.9%
4.1.x	Jelly Bean	16	6.8%
4.2.x		17	9.4%
4.3		18	2.7%
4.4	KitKat	19	31.6%
5.0	Lollipop	21	15.4%
5.1		22	20.0%
6.0	Marshmallow	23	10.1%

Figure 2.2: Android Platform Versions as at 30.05.2016[22]

whether they wanted to use the application, or not, and this has caused habituation even in later versions where permissions can be toggled[64], since users choose to approve permissions by reflex as part of the installation process of an application. Research has shown that 83% of application installations happen without the user consciously reading the permission list[28]. Android Marshmallow is only being run on 10.1% of all Android devices according the most current dashboard data at the time of writing[22], resulting in many users still being forced to follow the do-or-die model when installing applications(refer figure 2.2).

After Android Marshmallow

The current permission model assigns permissions into two categories; normal permissions and dangerous permissions(See Appendix 01 for a list of normal and dangerous permissions). Normal permissions are granted automatically when requested by a user, and can alter basic low-risk elements of a device's settings, such as the display brightness or wallpaper. Dangerous permissions are any permissions that can alter sensitive data on the device or use the device's higher-risk functions, such as connecting to the Internet, and are further classified into groups. If a permission from the same group has previously been approved by the user, then the permission will be automatically granted(i.e. if READ_SMS permission has been approved by the user then the device will not require permission to access the WRITE_SMS permission either). However, if a dangerous permission is in a category which has not been approved by the user earlier, a popup notification is generated at runtime, asking the user to approve the permission request. The permission, once granted, can be revoked by toggling a control in the application settings page later on. However, it should be noted that there has been criticism of the classification of permissions, and may researchers such as Felt have pointed out in earlier experiments that the classification should be based on risk to user privacy[26], resulting in classification systems that do not correlate with what is currently implemented

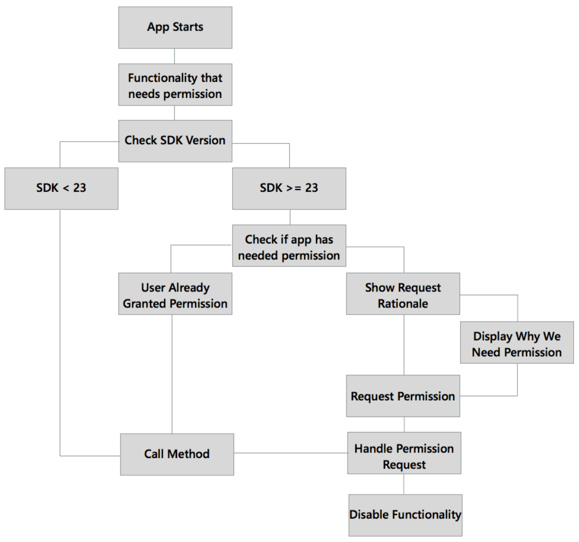

App Starts
Functionality that needs permission
Check SDK Version
SDK < 23
SDK >= 23
Check if app has needed permission
User Already Granted Permission
Show Request Rationale
Display Why We Need Permission
Request Permission
Call Method
Handle Permission Request
Disable Functionality

Figure 2.3: Android Permission Model- Flowchart[56]

by Google[6]. Refer figure 2.3 for a flowchart highlighting the recommended model for requesting permission in current Android devices.

2.1.2 Problems with the existing system

The current permission model is problematic and offers loopholes that can be manipulated to compromise privacy of user data. This section will include a condensed view of research in this domain which highlight and offer solutions to some of these issues.

Least Privilege

Both the most popular mobile operating systems, iOS and Android, require that developers follow the principle of "least privilege" with regard to permissions. The principle of least privilege states that "Every program and every user of the system should operate using the least set of privileges necessary to complete the job"[54]. In the context of Android applications, the principle states that developers should only ask for the very minimum of permissions that are required for the application to function[24]. Applications on the Apple AppStore are screened to check adherence to this principle(and other factors) before being made available for download[29]. However since there is no such screening process for Android applications, research has shown that these permission guidelines are not generally adhered to by developers[60]. Applications which do follow the permission guidelines are not necessarily popular with users, since permission requests either tend to be ignored due to lack of understanding[26] [38] or granted regardless of whether they are privacy sensitive or not due to habituation[28]. This results in many popular applications not following the principle of least privilege[63], with research showing that more than 33% end up asking for more permissions than are required[26].

Capability Leaks

Applications can sometimes access permissions which are not requested at install time. Such violations of the permission architecture to access data are referred to as 'capability leaks' [32] [31]. A tool named Woodpecker, which analyzes each application to detect readability of permissions from unguarded interfaces, is frequently used in research in this domain [70]. Through Woodpecker, two different types of capability leaks are identified; explicit leaks which find loopholes and access data without actually requesting permission and implicit leaks which let applications inherit permissions from another application, generally through intents. Other tools used to detect capability leaks include DroidChecker and IntentFuzzer [66] [17]. Capability leaks can also be exploited by malicious applications which use permissions which access permissions which have not been consiously granted to an application by a user for privilege escalation, using this to bypass restrictions on application functionality imposed due to the sand-boxing system[18].

Data Availability After Uninstallation

Since application permissions once granted are not revoked even upon uninstallation of an application, the data collected through the permissions granted while the application was installed may still be accessible once the app is uninstalled. Upon uninstallation, the user identity belonging to the application is deleted, but the permissions allowed are not revoked, and data still exists as orphans without a unique identifier (or parent). These orphans may later be exploited by malware causing privacy breaches and leaking of sensitive data [69]. Users misunderstanding or choosing not to read permission requests before granting create lasting consequences, the effects of which continue to compromise privacy even after uninstallation of the problematic applications.

Permission Creep

Permission creep occurs in applications that do not follow the principle of least privilege and ask for extra permissions[62]. Applications may sometimes require permissions that are not required for the core functionality of the application, but rather due to revenue generation methods, since most 'free' applications available on the PlayStore require in-app purchases for extra functionality. Some of these 'free' and low cost applications may sell data to advertisers to generate revenue, without explicit permission from the user. Extra permissions may also be requested in cases where developers have difficulty trying to align permission requests with the functionality required for the application, resulting in genuinely having to request extra permissions that seems unnecessary on analysis, but are mandatory for certain functions [62]. For example an update for the popular game Angry Birds caused controversy by requesting permission to send SMS messages, which is not part of the expected functionality of the application. However Rovio (the company behind Angry Birds) later explained that this is due to the payment methodology needed to purchase new levels, where an SMS message is sent to Rovio from the device to be billed later by the carrier [51] [47].

Studies have shown that over 50% of applications that request location access do so with the intent of sharing the information with advertisers for targeted marketing[53]. However, completely disallowing such requests would negatively impact the quality of applications available for Android since the revenue generation model would not survive. [46] In an ideal situation a user should be informed as to why an application is requesting a particular permission; as part of its core functionality, secondary functionality, as a method of revenue generation or any other usage for a permission to be requested. Research has shown that people tend to base their decisions on the reason behind data access[41]. However this is not possible with the current model since the level of information made available to users regarding application permission requests is decided on by the developer.

2.1.3 Proposed Solutions

Apart from tools developed to identify and provide solutions to the issues discussed above, there have been several research related to alternate models or methodologies that could be followed to mitigate privacy risks in the current model.

Facilitating Informed Decisions

Barerra et al. have suggested hierarchical solutions, recommending a methodology for emperical analysis of permission based security models using Self-Organizing Maps(SOMs), with the intention of identifying potential points of improvement for the current model while increasing information flow conveyed through the permission dialog without creating additional complexity[12]. Models to breakdown functionality needed by Android applications to create a more fine-grained permission toggle than is available currently has been developed, allowing users to have more control over what data is accessed by an application, and when it is used[15].

The current model does not allow users access to additional information as to why a particular permission is required, and studies have shown that this could be improved by analyzing human factors more effectively when allowing users to make such decisions. Research which focused on letting users choose applications based on their security and privacy expectations through including a "privacy facts" screen when downloading applications in addition to the list of permissions has proven that decisions made are different when users are given useful privacy information in addition to the current information provided[39]. Presenting users with personalized examples with their own data has also shown an increase in conscious decision making, and less habituation, for example by showing a sample of the users photos on the device and displaying "if you install this app, it will be able to access and delete the following of your photos" rather than just displaying the CAMERA permission[34].

Research has been carried out to design methods to recommend applications based on user's history of security and privacy concerns. Systems have been designed to track or reduce privacy violations by recommending applications based on users' security concerns[5]. Tools have also been developed to track information flow on a device to determine "tainted" information and predict privacy violations in real time[23]. Tools such as AppFence which give users the ability to track privacy violations and substitute fake data instead of actual user data for applications which violate privacy conditions have been developed, however this was not successful with all versions of Android and on average only worked for less than 33% of privacy violations without crashing[35].

Code Analysis to Curb Permission Creep

Applications requesting unecessary permissions and causing permission creep can be analyzed through static code analysis, by using tools such as pScout[10], and FlowDroid[9]. Static code analysis generally involves reading the Android Manifest and matching permissions which have been requested to those actually used by functions that provide core functionality of the application. Dynamic analysis of applications, runtime monitoring and Java code analysis have also been successful in identifying applications with permission creep [57]. Context of a permission request; time of the request, whether the screen is on or off, whether the application is running visibly as a background or foreground application or service, what the device was displaying while the permission request was taking place, frequency of repitive permission requests(for example searching for network or wifi information) have been shown to influence users when deciding whether a particular permission request is appropriate [64]. This conforms to Nissenbaum's Theory of Contextual Integrity[48] with regard to privacy, since the context and flow contribute towards the attitude of a user towards a privacy sensitive request, and not just information such as usage, reason etc.

2.1.4 Transitivity of Trust

Through this research we will attempt to create a web of trust for users on the Android platform. To our knowledge there are no existing research following this particular approach, however there are research that have been conducted applying PGP models and web of trust to situations which require reputation based trust assessment and user driven models of privacy protection.

Defining "Trust"

Researchers have different, sometimes contrasting definitions of "trust". McKnight et al. attempt to overcome issues in research in this domain that are caused due to conflicting definitions and a general lack of consensus about what "trust" entails. Comparing trust studies without a common definition is problematic, and the researchers have proposed two typologies for trust; a classification system for trust and definitions of six trust types to form a common model[44]. They define "Trusting Intention" corresponding to five factors:

- The prospect of negative consequences associated with trust
- Measure of dependence on another entity
- Feelings of security towards peers
- Context of trust and situation-specific factors
- Willingness to depend on another entity based on control

In our research we will be focusing on the dependence of trust on the context, and unexpectedness of permission requests in general. According to McKnight et al., complete trust is not possible in the human domain at once. Over time, a mental model of specific situations in which a particular entity can be trusted will be developed. This is applicable in the domain of Android permissions as well, as we will attempt to determine a method to value the trust accorded to peers by a user. Situational trust is defined as "the extent to which one intends to depend on a non-specific other party in a given situation", and research has suggested that this is stronger when the gain from trusting is numerically greater than the associated risk[37]. They suggest that situational trust is an intentional construct and relates to specific situations rather than generalizations.

Delhey et al. attempt to solve the radius of trst problem; how wide a circle of peers should be to incite trust, defining between "particular" and "general" trust[19]. They define trust circles for countries from different world regions, based on common factors and identify that in that situation the trust correlates to the product of the level of trust and the radius of trust, and suggest that research should discontinue using unspecified trust as a measure of general trust.

Definition of "Trust" in Computer Science

Serchan et al. have classified the concept of "trust" as it applies to the field of computer science, into two categories; user trust and system trust[55]. "User" trust is defined as being based on the evolution of a relationship based on the strength of user interactions through time, implying that trust is personalized. This is further sub-categorized into direct trust, where trust is based on direct interactions; and recommender trust, where trust is gained through evaluating experiences of peers with the entity in question. In contrast, "system" trust is derived from the security domain, and is defined as "the expectation that a device or system will faithfully behave in a particular manner to fulfill its intended purpose". In our research, both direct and recommender trust will be needed to evaluate users, and the users themselves have to have a measure of trust in the system in order for their recommendations of permissions to be valid.

Decentralization of Trust

Donvan et al. have conducted a comprehensive survey on trust models and their applications to the semantic web[8]. They have categorized models of trust proposed in previous research into 4 sections:

1. *Policy-based trust* - Models that are based on the assumption that trust is confirmed by obtaining a sufficient amount of credentials from a single reputable third party. Certificate authorities and other such methods for verifying trust through a centralized method are classified as policy-based trust models.

2. *Reputation-based trust* - These research propose models that use reputation and past actions and connections between entities to assess future behavior and enforce referral based trust or trust through first hand knowledge. Our proposed methods of using a web of trust classifies as a reputation based trust model.

3. *Generalized models of trust* - Other computable models of trust, some of which cannot be formally verified including research that take psychological factors into consideration when determining trust models are included in this section.

4. *Trust models in information resources* - Trust models developed to verify reliability of information on web sites are included in this section.

We will be focusing on research that have been conducted on reputation-based models of trust.

Application of Referral Trust in Other Systems

Research has been conducted into decentralization for reputation management in distributed systems, where peer referrals work to determine the reputation of an entity. Decentralization is viewed as being better than centralization in trust models for distributed systems which need to take human factors into consideration since "with decentralization, each rational entity will be allowed to take responsibility for its own fate, which is a basic human right"[4]. Sun et al. propose that in distributed mobile ad hoc networks, trustworthiness plays a role in predicting of and protecting against three types of malicious attacks and decentralized trust models can be used to verify authenticity of nodes rather than a central monitoring system[61].Combining reputation information from another peer, referred to as a "witness"[67][68], to make decisions by distributing reputation information has been followed in several research with decentralized reputation management, and approaches to combine information from an individual and other "witnesses" as a base for decision making in multi-agent systems has been discussed[52]. Research into computing degrees of trust in case other entities in the network being compromised result in false or conflicting information and application of trust to the peers themselves has been conducted, recommending methods to solve such conflicts in open systems[13].

Trust in Peer Networks

Trust networks have been used successfully to predict movie preferences of users from social network relationship data[30]. Golbeck et al. have defined trust in a person as "a commitment to an action based on a belief that the future actions of the person will lead to a good outcome", where the action and commitment does not have to be significant. The research assumes that user A trusts user B regarding user B is she chooses to act on a recommendation of user B. Similarly in our research we will be accepting a successful exchange of trust if a user chooses to follow peer recommendations on granting permissions to an application.

Massa et al. have researched collaborative filtering on recommended systems based on a trust network[42]. They use a nearest neighbor based approach to allow users to find resources based on the experience and opinions of their nearest neighbors. The issue of quality assessment of the web of trust that has been used, is solved through computing the reputation of users through trust propagation and performing similarity assessment to calculate number of successful recommendations and through that, evaluating the quality. An alternative method of letting users rate the feedback provided through the web of trust and using that rating as a measure of quality has also been suggested and is used in systems that use content filtering algorithms developed for online retailers such as Amazon and Ebay to recommend products to users, and has been proven successful in those scenarios. However it is questionable whether this method would be successful in rating application privacy decisions, as users themselves would not have prior expectations of privacy violations and would react differently than to a product recommendation.

Methods to allocate a value of "distrust" to a user by reversing the trust value or allocating a negative value have been researched previously, and comparisons have been made highlighting differences and similarities in the propagation of trust and distrust[33]. Distrust propogation has been applied to areas including web spam detection[65], detection of bullying and "trolls" in social networks[49] and modeling distrust among connections in a social network[71]. However, the implication of distrust among users will not be considered in our research at this stage.

Example: PageRank as a Web of Trust

One of the most commonly used and proven user driven privacy models is the "web of trust".Google's Page Rank algorithm is an implementation of this concept[50]. They define the web as a network of content without a centralized quality control methodology, and the authority of every page is inferred by the PageRank algorithm by examining the structure of the network. For example the global rank of a given page depends on the number and quality of incoming links. However PageRank is based on the link and not on a positive or negative association (i.e. if a reviewer negatively reviewed a product page and included a link to the page in question, then the PageRank algorithm would still count it as a valid incoming link when calculating popularity of a page).

Applying the concept used by PageRank to a user based network can be possible, with trust values that users cast on other users being used to predict trustworthiness of previously unknown users. Since trust is subjective(considering a user A, trust rating given to A by two different users may be entirely different) and asymmetric(A's rating for B is not equal to B's rating for A), the complete trust network should be considered when predicting trust among two individual users. Massa et al. divide trust into local and global metrics, and cite PageRank as a global metric since it approximates how the entire community as a whole rates the user in question[42]. Local metrics, which would have to be used in our implementation, are computationally more expensive, since they must be computed for each user separately.

2.2 Research Design

The final design will depend on the results of the survey and data collected from the modified PlayStore apk, and will be finalized at a later stage. The main objective of this research to analyze the viability of transitivity of trust which entails analyzing the extent to which people are open to trust other people in adopting their privacy preferences and quantifying the success of such an approach over the status quo. Finally we hope to implement required changes in the Android platform enabling users to adopt privacy preferences from people they trust.

2.3 Preliminary Results and Discussion

The preliminery survey included a list of questions based on the Internet Privacy Scale[14] to the extent to which users perceive their data as "private". Details of participants for the target group to form the network of trusted peers and the list of installed applications on their mobile phones, Android version, device information, and permissions granted/denied based on 10 popular applications from Google Play were collected through an ADB script which collects all the required data when a phone is connected to the machine. A random target group of users were chosen out of the respondents of the earlier survey for the user study.

Due to difficulty in locating compatible binaries for the devices we have on hand, we were unable to collect data as planned by asking participants to use a modified Android version for a selected time limit. A secondary survey was given with 12 questions based on the data collected from the field study[64]. Questions were in the format "You are using application A on your mobile phone while application B and C are running in the background. Application D asks for X permission. What would be your response in this scenario?" with the choice to either grant or deny a permission request. Participants were also asked to name a "trusted peer" who they would trust to make privacy decisions for them(a sample list of questions are included in Appendix 02). Data collection from the survey is partially complete. Graphs generated from the survey are shown in figure 2.5.

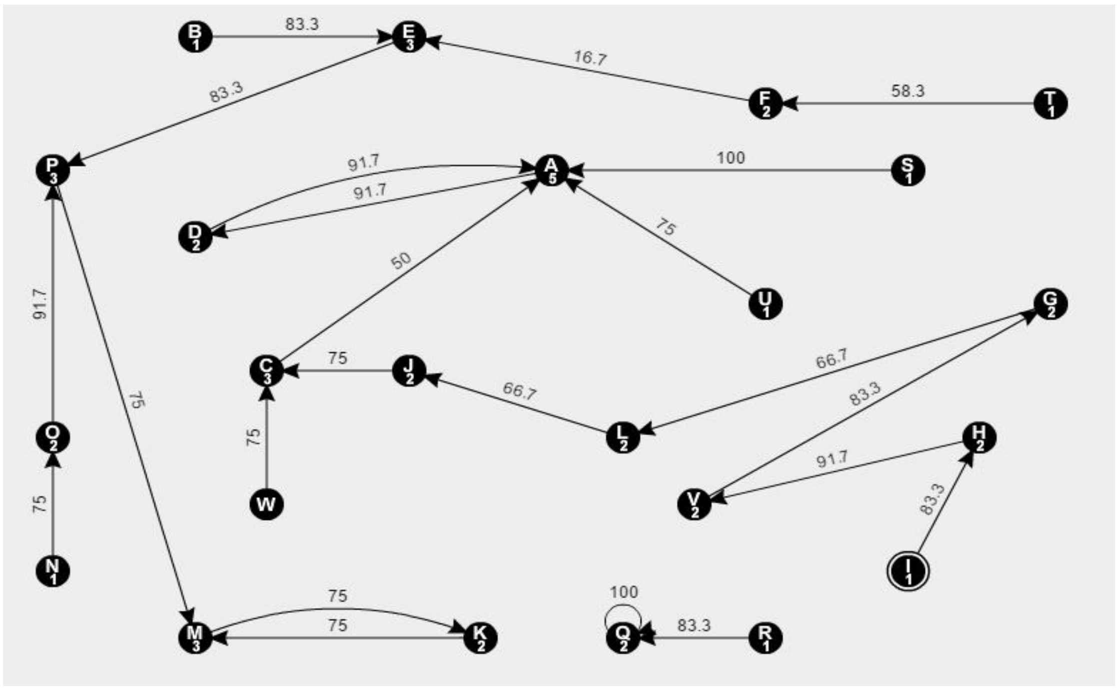
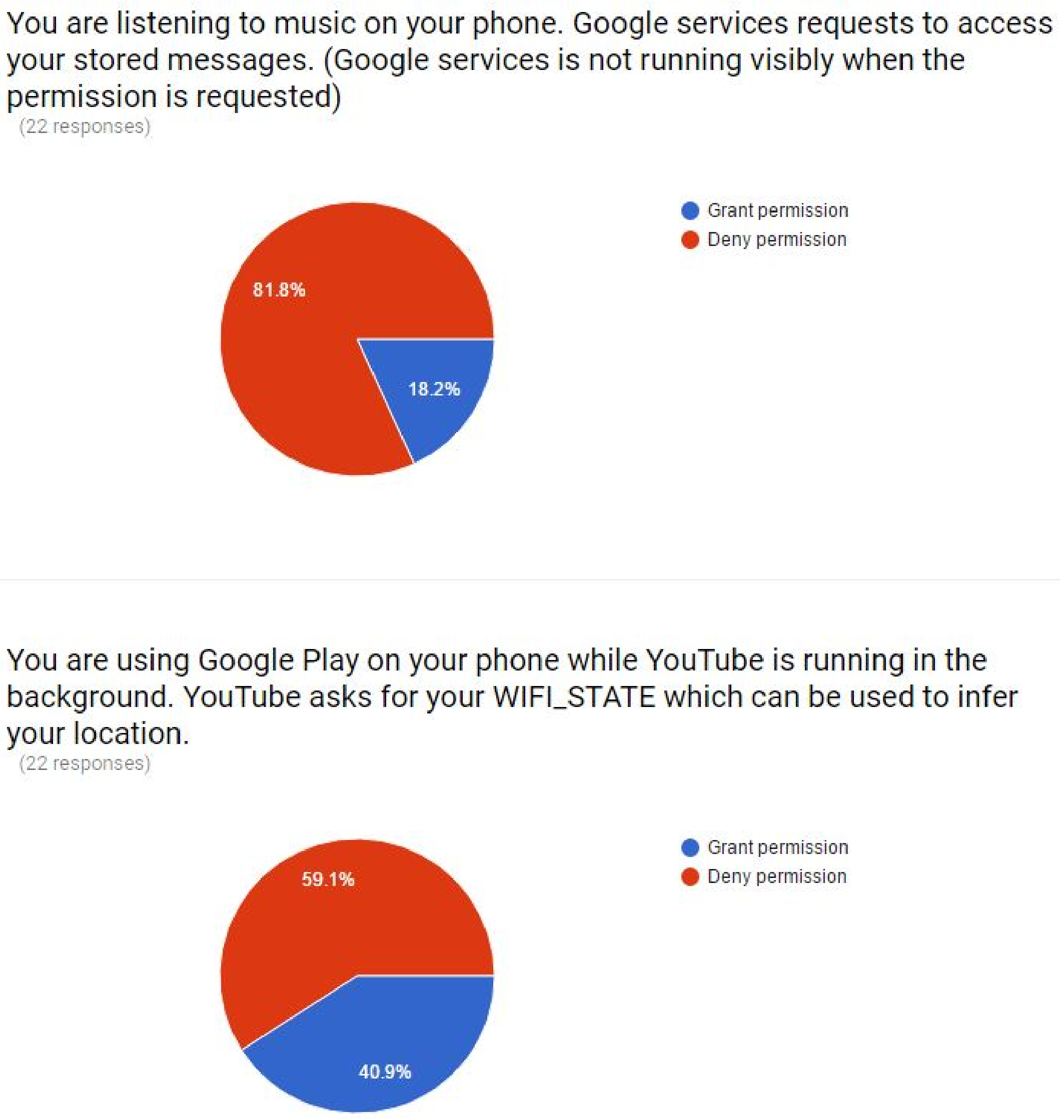


Figure 2.4: Partial representation of trusted peers among survey respondents

The partial data was represented using a graph(Figure 2.4). Participants were each assigned a letter of the alphabet. A directed edge from A to B denotes that A named B as a trusted peer, and the weight of an edge represents the percentage of similarity between the privacy choices of A and B. Each response was assigned a binary value(1 for "grant permission" and 0 for "deny permission") and represented as a string, and the percentage of similarity was calculated using a Python string matching script. The weight assigned to each edge is represented as a percentage for ease of understanding. For example the edge from B-¿E denotes that B named E as a trusted peer, and B's privacy preferences match those of E 83.3% of the time. Nodes with a higher indegree are people who are more "trusted" by their peers. Since each participant was asked to name *one* "peer" all nodes have an outdegree of 1.

The completed graph can be used as a measure to evaluate the difference in expectations and reality in terms of "trust transitivity" among the target group. Apart from the trust rating between E and F (16.7%) all other responses show trust similarity among peers of 50% or more. The average matching percentage is **75.72%**. Since trust is subjective, it can be argued that the results are not conclusive in terms of similarity as even "peers" with a 100% similarity may choose different privacy options depending on the context in which they occur. However the current permission model does not offer any reassurance that the permission decisions of users do not lead to privacy violations, as the context in which a granted permission is requested and utilized is not fully explained. Research[64] conducted to determine the similarity between user expectations



You are listening to music on your phone. Google services requests to access your stored messages. (Google services is not running visibly when the permission is requested)

(22 responses)

You are using Google Play on your phone while YouTube is running in the background. YouTube asks for your WIFI_STATE which can be used to infer your location.

(22 responses)

Figure 2.5: Some responses from the survey

	Success Rate
Ask-On-Install	25%
Marshmallow	80%
Our approach	75.72%

Table 2.1: Comparison of Partial Results

when granting a permission and privacy violations which occur has shown that in versions before Marshmallow, user expectations are met only 25% of the time, while there is an improvement to 80% in newer versions of Android. However only 15.2% of users have upgraded to devices with Android versions newer than Marshmallow[22]. Compared to the success rate of the current install time model(25%) the result shown from our model of 75% is a 300% improvement on the previous system(see table 2.1).

User habituation due to increased user involvement in privacy decision making is a key issue in Android versions newer than Marshmallow as well. Therefore even though Marshmallow has 80% success rate it still requires extensive user involvement. However our approach significantly reduces the user habituation by letting the user adopt privacy preferences of someone they trust. In later stages of the research we hope to quantify the actual level of user involvement as well. There are additional factors to be considered such as the feasibility of adapting permission requests in every context, and applicability according to device versions. Our user group consisted of UCSC students who are relatively well-informed about privacy violations. Further analysis is required before proceeding with the proposed modifications to PlayStore apk which will enable users to choose to adopt privacy settings of more experienced peers and make more informed decisions about permission requests, and enable us to log actual user involvement and decrease in problems caused due to habituation.

Appendix 01 - Normal and Dangerous Permissions

Normal Permissions

ACESS_LOCATION_EXTRA_COMMANDS

ACCESS_NETWORK_STATE

ACCESS_NOTIFICATION_POLICY

ACCESS_WIFI_STATE

BLUETOOTH

BLUETOOTH_ADMIN

BROADCAST_STICKY

CHANGE_NETWORK_STATE

CHANGE_WIFI_MULTICAST_STATE

CHANGE_WIFI_STATE

DISABLE_KEYGUARD

EXPAND_STATUS_BAR

GET_PACKAGE_SIZE

INSTALL_SHORTCUT

INTERNET

KILL_BACKGROUND_PROCESSES

MODIFY_AUDIO_SETTINGS

NFC

READ_SYNC_SETTINGS

READ_SYNC_STATS

RECEIVE_BOOT_COMPLETED

REORDER_TASKS

REQUEST_IGNORE_BATTERY_OPTIMIZATIONS

REQUEST_INSTALL_PACKAGES

SET_ALARM

SET_TIME_ZONE

SET_WALLPAPER

SET_WALLPAPER_HINTS

TRANSMIT_IR

UNINSTALL_SHORTCUT

USE_FINGERPRINT

VIBRATE

WAKE_LOCK

WRITE_SYNC_SETTINGS

Dangerous Permissions

READ_CALENDAR

WRITE_CALENDAR

CAMERA

READ_CONTACTS

WRITE_CONTACTS

GET_ACCOUNTS

ACCESS_FINE_LOCATION

ACCESS_COARSE_LOCATION

RECORD_AUDIO

READ_PHONE_STATE

CALL_PHONE

READ_CALL_LOG

WRITE_CALL_LOG

ADD_VOICEMAIL

USE_SIP

PROCESS_OUTGOING_CALLS

BODY_SENSORS

SEND_SMS

RECEIVE_SMS

READ_SMS

RECEIVE_WAP_PUSH

RECEIVE_MMS

READ_EXTERNAL_STORAGE

WRITE_EXTERNAL_STORAGE

Appendix 02 - Sample Questions

You are listening to music on your phone. Google services requests to access your stored messages.(Google services is not running visibly when the permission is requested)

You are using Google Play on your phone while YouTube is running in the background. YouTube asks for your WIFI_STATE which can be used to infer your location.

You are using Google Play on your phone while Facebook is running in the background. Facebook queries your location.

You are using the web browser on your device. It asks for your WIFI_STATE which could be used to infer your location.

You are using Google Play on your phone. Instagram asks for your location. (Instagram is not running visibly when the permission is requested)

You are using Google Search on your device. The Yahoo Mail application asks for your WIFI_STATE which could be used to infer your location.

You are using Google Chrome on your mobile device. The Avast Antivirus application accesses your browsing history.

You are not using your device. It is switched on and Google Play is the application running in the background. It asks for your location.

You are not using your device. It is switched on but no apps are running visibly. Your weather application asks for your location.

You are using Google Play on your device. The TrueCaller application accesses your call history.

You are using Google Search on your device. Your messaging application sends a sms.

You are using your device, but no specific apps are running. Android.Contacts application accesses your location.

Bibliography

[1] The android open source project. `https://source.android.com/`.

[2] Patterns: Permissions. `https://www.google.com/design/spec/patterns/permissions.html#`.

[3] Market share statistics for internet technologies; mobile/tablet operating system market share. `https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1`, 2016.

[4] Alfarez Abdul-Rahman and Stephen Hailes. Using recommendations for managing trust in distributed systems. In *Proceedings IEEE Malaysia International Conference on Communication*, volume 97. Citeseer, 1997.

[5] Hussain MJ Almohri, Danfeng Daphne Yao, and Dennis Kafura. Droidbarrier: Know what is executing on your android. In *Proceedings of the 4th ACM conference on Data and application security and privacy*, pages 257–264. ACM, 2014.

[6] AOSP. Android security 2015 year in review - android open source project. `https://static.googleusercontent.com/media/source.android.com/en//security/reports/Google_Android_Security_2015_Report_Final.pdf`.

[7] AOSP. Security program overview. `https://source.android.com/security/`.

[8] Donovan Artz and Yolanda Gil. A survey of trust in computer science and the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):58–71, 2007.

[9] Steven Arzt, Siegfried Rasthofer, Christian Fritz, Eric Bodden, Alexandre Bartel, Jacques Klein, Yves Le Traon, Damien Octeau, and Patrick McDaniel. Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps. In *ACM SIGPLAN Notices*, volume 49, pages 259–269. ACM, 2014.

[10] Kathy Wain Yee Au, Yi Fan Zhou, Zhen Huang, and David Lie. Pscout: analyzing the android permission specification. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 217–228. ACM, 2012.

[11] Benjamin Aziz and Mario Tejedor-Gonzalez. An android pgp manager: towards bridging end-user cryptography to smart phones. *International Journal of Security*, 6(5), 2012.

[12] David Barrera, H Güneş Kayacik, Paul C van Oorschot, and Anil Somayaji. A methodology for empirical analysis of permission-based security models and its application to android. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 73–84. ACM, 2010.

[13] Thomas Beth, Malte Borcherding, and Birgit Klein. Valuation of trust in open networks. In *European Symposium on Research in Computer Security*, pages 1–18. Springer, 1994.

[14] Tom Buchanan, Carina Paine, Adam N Joinson, and Ulf-Dietrich Reips. Development of measures of online privacy concern and protection for use on the internet. *Journal of the American Society for Information Science and Technology*, 58(2):157–165, 2007.

[15] Sven Bugiel, Stephan Heuser, and Ahmad-Reza Sadeghi. Flexible and fine-grained mandatory access control on android for diverse security and privacy policies. In *Usenix security*, pages 131–146, 2013.

[16] Windows Dev Center. Msdn, the app certification process. `https://msdn.microsoft.com/en-us/windows/uwp/publish/the-app-certification-process`.

[17] Patrick PF Chan, Lucas CK Hui, and Siu-Ming Yiu. Droidchecker: analyzing android applications for capability leak. In *Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks*, pages 125–136. ACM, 2012.

[18] Lucas Davi, Alexandra Dmitrienko, Ahmad-Reza Sadeghi, and Marcel Winandy. Privilege escalation attacks on android. In *Information Security*, pages 346–360. Springer, 2010.

[19] Jan Delhey, Kenneth Newton, and Christian Welzel. How general is trust in most people? solving the radius of trust problem. *American Sociological Review*, 76(5):786–807, 2011.

[20] Android Developers. Android developers: Dashboards. `http://developer.android.com/about/dashboards/index.html`.

[21] Android Developers. Publishing overview. `http://developer.android.com/tools/publishing/publishing_overview.html`.

[22] Android Developers. Android dashboards. `https://developer.android.com/about/dashboards/index.html`, 2016.

[23] William Enck, Peter Gilbert, Seungyeop Han, Vasant Tendulkar, Byung-Gon Chun, Landon P Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N Sheth. Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS)*, 32(2):5, 2014.

[24] William Enck, Machigar Ongtang, and Patrick McDaniel. Understanding android security. *IEEE security & privacy*, (1):50–57, 2009.

[25] Patrick Feisthammel. Pgp: Explanation of the web of trust of pgp. `https://www.rubin.ch/pgp/weboftrust.en.html`, 2004.

[26] Adrienne Porter Felt, Erika Chin, Steve Hanna, Dawn Song, and David Wagner. Android permissions demystified. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 627–638. ACM, 2011.

[27] Adrienne Porter Felt, Kate Greenwood, and David Wagner. The effectiveness of application permissions. In *Proceedings of the 2nd USENIX conference on Web application development*, pages 7–7, 2011.

[28] Adrienne Porter Felt, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner. Android permissions: User attention, comprehension, and behavior. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, page 3. ACM, 2012.

[29] Peter Gilbert, Byung-Gon Chun, Landon P Cox, and Jaeyeon Jung. Vision: automated security validation of mobile apps at app markets. In *Proceedings of the second international workshop on Mobile cloud computing and services*, pages 21–26. ACM, 2011.

[30] Jennifer Golbeck. Generating predictive movie recommendations from trust in social networks. In *International Conference on Trust Management*, pages 93–104. Springer, 2006.

[31] Michael Grace, Yajin Zhou, Zhi Wang, and Xuxian Jiang. Detecting capability leaks in android-based smartphones. Technical report, Technical Report North Carolina State University, 2011.

[32] Michael C Grace, Yajin Zhou, Zhi Wang, and Xuxian Jiang. Systematic detection of capability leaks in stock android smartphones. In *NDSS*, 2012.

[33] Ramanthan Guha, Ravi Kumar, Prabhakar Raghavan, and Andrew Tomkins. Propagation of trust and distrust. In *Proceedings of the 13th international conference on World Wide Web*, pages 403–412. ACM, 2004.

[34] Marian Harbach, Markus Hettig, Susanne Weber, and Matthew Smith. Using personal examples to improve risk communication for security & privacy decisions. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 2647–2656. ACM, 2014.

[35] Peter Hornyack, Seungyeop Han, Jaeyeon Jung, Stuart Schechter, and David Wetherall. These aren't the droids you're looking for: retrofitting android to protect data from imperious applications. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 639–652. ACM, 2011.

[36] Apple Inc. ios developer library-app distribution guide. `https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/SubmittingYourApp/SubmittingYourApp.html`.

[37] Herbert W Kee and Robert E Knox. Conceptual and methodological considerations in the study of trust and suspicion. *Journal of Conflict Resolution*, pages 357–366, 1970.

[38] Patrick Gage Kelley, Sunny Consolvo, Lorrie Faith Cranor, Jaeyeon Jung, Norman Sadeh, and David Wetherall. A conundrum of permissions: installing applications on an android smartphone. In *Financial Cryptography and Data Security*, pages 68–79. Springer, 2012.

[39] Patrick Gage Kelley, Lorrie Faith Cranor, and Norman Sadeh. Privacy as part of the app decision-making process. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3393–3402. ACM, 2013.

[40] Alex Kinsella. How to submit your apps for blackberry 10. `http://devblog.blackberry.com/2012/10/submit-apps-blackberry-10/1`.

[41] Jialiu Lin, Bin Liu, Norman Sadeh, and Jason I Hong. Modeling users mobile app privacy preferences: Restoring usability in a sea of permission settings. In *Symposium On Usable Privacy and Security (SOUPS 2014)*, pages 199–212, 2014.

[42] Paolo Massa and Paolo Avesani. Trust-aware collaborative filtering for recommender systems. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 492–508. Springer, 2004.

[43] Patrick McDaniel. Bloatware comes to the smartphone. *IEEE Security & Privacy*, (4):85–87, 2012.

[44] D Harrison McKnight and Norman L Chervany. The meanings of trust. 1996.

[45] PD Meshram and RC Thool. A survey paper on vulnerabilities in android os and security of android devices. In *Wireless Computing and Networking (GCWCN), 2014 IEEE Global Conference on*, pages 174–178. IEEE, 2014.

[46] T. Moynihan. Wired: Apps snoop on your location way more than you think. `http://www.wired.com/2015/03/apps-snoop-location-way-think/`, 2015.

[47] P. Nickinson. Android central: Rovio explains new permissions in angrybirds seasons update. `http://www.androidcentral.com/rovio-explains-newpermissions-angry-birds-seasons-update`, 2014.

[48] Helen Nissenbaum. Privacy as contextual integrity. *Wash. L. Rev.*, 79:119, 2004.

[49] F Javier Ortega, José A Troyano, FermíN L Cruz, Carlos G Vallejo, and Fernando EnríQuez. Propagation of trust and distrust for the detection of trolls in a social network. *Computer Networks*, 56(12):2884–2895, 2012.

[50] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: bringing order to the web. 1999.

[51] A. Russakovskii. Rovio explains why the sms permission was introduced in angry birds v1.5.1. `http://www.androidpolice.com/2011/02/06/rovio-explainswhy-the-sms-permission-was-introduced-in-angry-birds-v1-5-1/`, 2011.

[52] Jordi Sabater and Carles Sierra. Reputation and social network analysis in multi-agent systems. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 475–482. ACM, 2002.

[53] N Saint. 50% of android apps with internet access that ask for your location send it to advertisers, 2010.

[54] Fred B Schneider. Least privilege and more. In *Computer Systems*, pages 253–258. Springer, 2004.

[55] Wanita Sherchan, Surya Nepal, and Cecile Paris. A survey of trust in social networks. *ACM Computing Surveys (CSUR)*, 45(4):47, 2013.

[56] Mobile Siri. `http://mobilesiri.com/how-to-request-android-runtime-permissions/`, 2016.

[57] Michael Spreitzenbarth, Felix Freiling, Florian Echtler, Thomas Schreck, and Johannes Hoffmann. Mobile-sandbox: having a deeper look into android applications. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pages 1808–1815. ACM, 2013.

[58] Statistica. Number of apps available in leading app stores as at july 2015. `http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/`, 2016.

[59] Gunnar Stevens and Volker Wulf. Computer-supported access control. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 16(3):12, 2009.

[60] Ryan Stevens, Jonathan Ganz, Vladimir Filkov, Premkumar Devanbu, and Hao Chen. Asking for (and about) permissions used by android apps. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, pages 31–40. IEEE Press, 2013.

[61] Yan Sun, Zhu Han, and KJ Ray Liu. Defense of trust management vulnerabilities in distributed networks. *IEEE Communications Magazine*, 46(2):112–119, 2008.

[62] Timothy Vidas, Nicolas Christin, and Lorrie Cranor. Curbing android permission creep. In *Proceedings of the Web*, volume 2, pages 91–96, 2011.

[63] Xuetao Wei, Lorenzo Gomez, Iulian Neamtiu, and Michalis Faloutsos. Permission evolution in the android ecosystem. In *Proceedings of the 28th Annual Computer Security Applications Conference*, pages 31–40. ACM, 2012.

[64] Primal Wijesekera, Arjun Baokar, Ashkan Hosseini, Serge Egelman, David Wagner, and Konstantin Beznosov. Android permissions remystified: a field study on contextual integrity. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 499–514, 2015.

[65] Baoning Wu, Vinay Goel, and Brian D Davison. Propagating trust and distrust to demote web spam. *MTW*, 190, 2006.

[66] Kun Yang, Jianwei Zhuge, Yongke Wang, Lujue Zhou, and Haixin Duan. Intentfuzzer: detecting capability leaks of android applications. In *Proceedings of the 9th ACM symposium on Information, computer and communications security*, pages 531–536. ACM, 2014.

[67] Bin Yu and Munindar P Singh. A social mechanism of reputation management in electronic communities. In *International Workshop on Cooperative Information Agents*, pages 154–165. Springer, 2000.

[68] Bin Yu and Munindar P Singh. An evidential model of distributed reputation management. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 294–301. ACM, 2002.

[69] Xiao Zhang, Kailiang Ying, Yousra Aafer, Zhenshen Qiu, and Wenliang Du. Life after app uninstallation: Are the data still alive? data residue attacks on android. 2016.

[70] Yajin Zhou, Zhi Wang, Wu Zhou, and Xuxian Jiang. Hey, you, get off of my market: Detecting malicious apps in official and alternative android markets. In *NDSS*, 2012.

[71] Cai-Nicolas Ziegler and Georg Lausen. Propagation models for trust and distrust in social networks. *Information Systems Frontiers*, 7(4-5):337–358, 2005.