# XXXXX

XXXXXX
Rochester Institute of Technology, Rochester, NY, USA
{xxxx,xxxxx,xxxxx}@rit.edu

## ABSTRACT

Android applications (apps) rely on a permission-based model to carry out core functionality. The way that Android apps ask a user to accept permissions has recently changed from being an all or nothing accept or reject upon installation the application, to now asking for permission at runtime. A primary goal of this change was to provide users more control of the permissions their apps utilize. Do users actually feel more comfortable with this change? Are they more knowledgeable about the permissions their apps are using?

We conducted a large, in person study involving more than 330 participants from a diverse background set to determine if this new permission model makes users feel more comfortable and knowledgable about the permissions their apps are using. We also sought to determine what impact providing a rationale had on user's accepting permission requests.

We found that.....

## Keywords

XXXXXX

## 1. INTRODUCTION

Android has become the world?s most popular mobile platform, allowing users to perform a variety of tasks that were previously unachievable in a mobile environment. Android applications (apps) provide the user with a diverse set of functionality allowing them to do everything from posting their Facebook status, to performing banking transactions. In order to perform various sensitive tasks, an app must be granted appropriate permissions to carry out this functionality. For example, if an app requests access to a user's contact list, the app much explicitly ask the user for permission to access this information. These app permissions serve as as an integral component of the app's security by limiting access to this information and functionality to other potentially malicious apps installed on the user's device, but also in ensuring that a user is adequately able to protect the information on their phone as they desire.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

In Since its inception, Android has used a system where users were asked to accept all of an app's permissions at install time. If they did not approve all of an app's requested permissions, they would not be allowed to install or use the app. Numerous works stated issues with this design from numerous perspectives ranging from security issues since a user might not perform an important security update to an app since it requested a permission they did not approve of to more simple usability issues. **[cite]**

The manner in which apps requested permissions substantially changed with the introduction of Android Marshmallow (Android M), which adopted a process of an app asking for permissions at runtime, and not install time. Users are now able to choose to provide an app specific functionality while still installing and using an app (without the functionality afforded by the permission), and even change the app's permission settings whenever they pleased, even long after installation.

The conversion to this new permission structure had several intentions including making users feel more comfortable, streamline the app installation process, and provide them more authority over their data and privacy, and allow for automatic updates since a user would no longer be required to accept all changed permissions at install time [1].

We sought to determine if these goals were being met by performing a large study involving 321 participants from diverse backgrounds. Our primary research questions were: **[add in RQs]**

## 2. BACKGROUND & RELATED WORK

### 2.1 Android Permissions

Android apps operate under a privilege-separated system where each app operates with a specific system identify, and each app is isolated from all other apps on the device. More fine-grain functionally enforces permissions on specific operations which the app may carry out. For example, if an app wishes to access the internet, or the user's contacts, the app will be required to be granted this permission before the app has the ability to access this functionality. Permissions are separated into several protection levels, with the most important being *normal* and *dangerous* permissions [2].

1. **Normal**: These permissions cover functionality that the app needs which are located outside of its sandbox, but are deemed to create very little risk to the user's privacy, or to other apps. An app that requests a normal permissions is automatically granted this function-

ality. In Android Marshmallow, some normal permissions include `BLUETOOTH`,`ACCESS_WIFI_STATE` and `INTERNET`.

2. **Dangerous**: These permissions are deemed to pose significant risks to the user's privacy, or to other apps installed on the device. The user must explicitly grant access to any dangerous permissions requested by the app. In Android Marshmallow, some dangerous permissions include `READ_CALENDAR`, `READ_SMS` and `CAMERA`. In Android Marshmallow, all dangerous permissions belong to a specific permission group. For example, all dangerous permissions which are related to phone calls belong to the *Phone* group. When an app requests a dangerous permission in a group which the user has already granted a permission in, the app automatically gains the ability to use that permission since the app has already been granted a permission in that group. For example, if the app requests the `CALL_PHONE` permission, but the app has already been granted the `ADD_VOICEMAIL` permission, the app will automatically gain access to the `CALL_PHONE` permission since they both reside in the same group.

Prior to Android Marshmallow, the user was prompted to accept or reject all dangerous app permissions whenever installing an app. When updating an app, they were also prompted to accept or reject all new permissions the app was requesting. If the user desired to not approve even a single permission, they would not be allowed to update install the app. It was an all other nothing ordeal. Users were also not able to change permissions after installation and the only way to restrict an app's access to a specific permission was to uninstall the entire app [7]. In the past several years, there has been a substantial amount of research recommending changes to the Android permission model ranging from the creation of privacy profiles [4] to more granular sets of Android permissions [6].

Marshmallow represented a significant change with how permissions were handled in Android apps. Users would now be prompted about allowing an app's permission requests at runtime, and not upon installing the application. The intention is that this would make installing and updating the app a simpler, easier process while providing the user greater access over the app, and therefore their privacy. The user would also be able to use the app, but choose the permissions they wanted to allow it to have, and even change their permission decisions whenever they pleased. The developer would also have the option of providing a *rationale*, or further explanation, to the user about why the app was asking for a particular permission.

## 2.2 Related Work

# 3. METHODOLOGY

– Introduce user study & hypothesis

## 3.1 Hypothesis

Upon beginning our work, we began with several hypothesis:

1. **Marshmallow helps users feel more comfortable using the apps**: The new permission model affords users greater control to choose the permissions that their app uses. We believe that this will make users feel more comfortable with using the apps.
2. **Permissions in Android M are easier to understand and recall if we add meaningful rationale to the requested permissions**: (find external works to back this up or support it) - **[Find reasons to back up the way I believe that I do]**
3. **Most users still have poor comprehension of Android permissions**: Previous research has also shown that the majority of users do not properly comprehend what permissions actually mean [3]. We believe that our findings for both the Pre-M and Marshmallow versions of the app will support these previous findings. **[add more citations]**
4. **Providing a rationale will help users feel more comfortable with accepting permissions**: XXXXX**[find rationale]**
5. **Most users will continue to pay little attention to permissions**: Previous research conducted on the Pre-M permission model has shown that most users do not pay attention to permissions when installing an app [3]. Although the new Marshmallow model will provide users with more control, and in many ways make them more aware of the permissions they are installing, we still believe that permission attention will continue to be a problem.

## 3.2 Study Design

Our study was comprised of three primary components: User Recruitment, the Tic-Tac-Toe app, and Questionnaire, which we will describe.

### 3.2.1 *Recruitment*

Our human study was conducted at Imagine RIT[1], a single day event where thousands of members of the community visit the RIT campus and view a variety of scientific and educational venues ranging from robotics, software projects, and engineering activities. With the assistance of student volunteers, we set up two tables with six MacBook laptops running an Android emulator. Two laptops were running the emulator with API 22 (Pre-M), while the other four were running API 23 (Marshmallow). Participants were recruited by asking visitors passing by our table if they would like to participate in an Android study. Users were only provided vague details about the study being about Android permissions in the event pamphlet.

### 3.2.2 *Application*

We developed a simple tic, tac, toe app for use in our study, which contained the following permissions: `ACCESS_FINE_LOCATION`, `GET_ACCOUNTS` and `READ_PHONE_STATE`. We selected these permissions since they have been identified as commonly used permissions in Ad libraries [5]. When beginning to use the app, users were told to act like they were using the app on their personal device and were randomly asked to participant using one of three nearly identical versions of the app:

1. **Android Pre-Marshmallow** (API 22): Users were asked to accept or deny all app permissions at install time, a process which replicates how Android has asked users for permissions since its inception.

[1]http://www.rit.edu/imagine/

2. **Android Marshmallow - No Rationale** Users were asked to accept or reject permissions at runtime. At the beginning of the game, participants were told that the app needed to locate people around them and the user was asked if they wanted to allow the app to access the device's location. After playing the game, the app asked for the permissions to make calls and access the user's contacts.

3. **Android Marshmallow - With Rationale** (API 23): This version of the app was identical to the No Rationale version, except that a rationale was provided to the user before being prompted for their permission. For example, before asking for permission to access the user's contacts, a rationale stated that "This App needs to access your contacts to share results."

Both Marshmallow versions of the app were connected to a database which recorded whenever a user accepted or denied a permission. Since all permission requests are made at installation for Pre-M apps, there was no data to record for this API level.

The app began by asking users to start a new game, and was followed by participants selecting several area participants to play against. Since the primary objective of the research was to understand how users react to the new permission model, we felt comfortable mocking out this functionality in the app. The participants would then play the game and were notified of their victory or defeat at the conclusion of the game. Once the game was completed, users were asked to share their results with the contacts on their device.

### 3.2.3 *Questionnaire & Exit Survey*

Before using the app, participants were initially asked to provide basic information about themselves such as age, gender, highest level of education, and how long they have used Android. Participants were not required to have ever used Android. After playing the game, participants were then asked to fill out the second portion of the questionnaire, which was nearly identical for all three of the user groups, except for certain questions pertaining to each group such as how helpful the permission rationale was, or if the user felt comfortable with accepting or rejecting permissions at runtime. Users were not allowed to go back and use the game to assist them in answering any of the questions, and were told to do the best they could in accurately answering all questions.

## 4. RESULTS

Our work was driven by XXX primary research questions:

**RQX: What effect does the new permission model have on a user's ability to recall an app's permissions?**

**[find citations about why it is important for users to know the permissions in their app]**

A primary goal of the new permission model was for users to be more knowledgable about the permissions their apps were using. As part of our user study, we asked users how easy it was to recall the permissions they accepted and rejected

**[Pradeep: Get the numbers for this]**

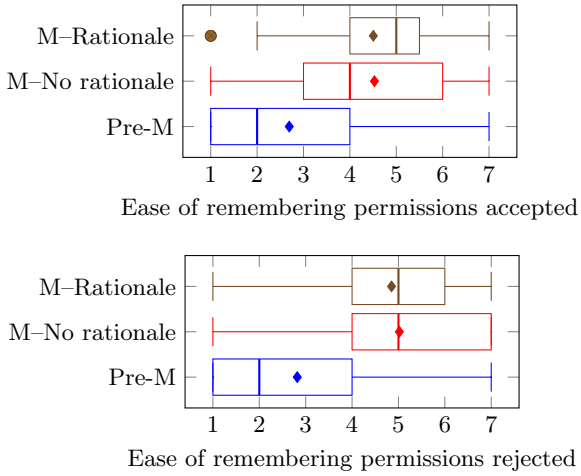


Figure 1: Ease with which participants remember the permissions they accepted and rejected.

We next sought to determine if users were actually able to recall the permissions the app requested. As part of our post activity survey, we asked to users to select all the permissions the app used from a list of six possible options. We then measured the precision, recall and F-score of the user's ability to correctly recall the permissions the app had used. The results of this analysis are shown in Figure 2.

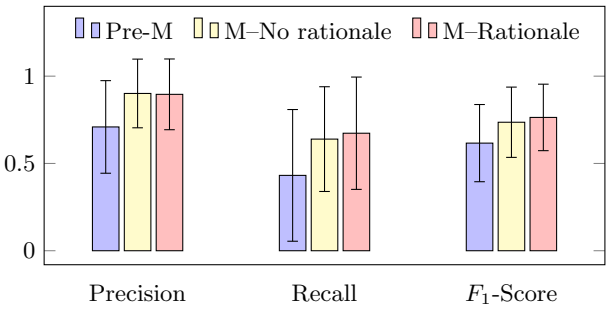


Figure 2: Participants' ability to remember permissions.

There was only a very small difference in the results between the 'M - With Rationale' and 'M - Without Rationale' groups, which is not surprising since the same permission requests were still shown at the same time to the users, with only an extra reasoning dialog shown to the with rationale group. However, there was a large difference in the three evaluated groups between the two 'M' groups, and the 'Pre-M' group. This demonstrates that users were able to more accurately able to recall the proper requested permissions for each of the Marshmallow groups in comparison to the Pre-M group. With the Pre-M model, users are shown the app's permissions at install time. Very frequently, users would quickly skim through this list in an attempt to use the app as soon as possible. Users also voiced their displeasure with the permission model in their feedback, with one user stating: "Either you accept the condition of permission or you cannot install the application. that's a really annoying thing. I'd like to know why a particular app needs the permissions it needs." This lack of attention to the permissions

in the Pre-M model is supported by previous research which demonstrates that few users pay attention to permissions when installing an app and that very few users are able to correctly recall the permissions an app uses [3].

This demonstrates that

**[See if there is a correlation between the ability to recall permissions and if users said it was easy - Do on an individual level]**

**Discussion:**

**RQX:** What effect does the rationale have on a user's perception of a permission request?

We next sought to determine the effect that providing a meaningful rationale had on a user accepting or denying a permission request. In order to explore this question, we provided users with nearly two identical versions of the same app, with the only difference being that one version provided users with a meaningful rationale as to why the permission was required. Whenever a user accepted or rejected a permission request, we recorded these results in a remote database. In the 'With Rationale' version of the app, we provided a meaningful rationale as to why the app was requesting the permission before asking the user for their location (`ACCESS_FINE_LOCATION`) and if the app could have access to the contacts information on the device (`GET_ACCOUNTS`). In order to act as a control between the two groups, we provided no rationale for the `READ_PHONE_STATE` permission. *[Dan says: best way to say this?]* The results of this analysis are shown in Table 1.

Table 1: Permission Model User Acceptance

| Permission | Rat Accept | No RatAccept |
|---|---|---|
| **ACCESS_FINE_LOCATION** | 0.95 | 0.86 |
| **GET_ACCOUNTS** | 0.62 | 0.49 |
| **READ_PHONE_STATE** | 0.59 | 0.57 |

These results demonstrate how a meaningful rationale can influence a person to accept

Although we have found meaningful results, we believe that there is substantial room for future research in this area. For example, **[talk about the future work]**

**Discussion:**

**RQX:** Which model helps users feel more Comfortable?

# 5. OTHER OBSERVATIONS

# 6. LIMITATIONS & FUTURE WORK

# 7. CONCLUSION

## Acknowledgment

We would like to thank the following students for their contributions to the project: Hussein Talib, Cesar Fernandez, Silva Matti, Paula Garcia, Chris Lentner, Daniel Santoro, Taylor Corrello, Jodie Miu, George Hearde, and Kocsen Chung. This work was partially funded through a development grant from RIT

# 8. REFERENCES

[1] Requesting permissions at run time. https://developer.android.com/training/permissions/requesting.html.

[2] System permissions. https://developer.android.com/guide/topics/security/permissions.html.

[3] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner. Android permissions: User attention, comprehension, and behavior. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, SOUPS '12, pages 3:1–3:14, New York, NY, USA, 2012. ACM.

[4] B. Liu, J. Lin, and N. Sadeh. Reconciling mobile app privacy and usability on smartphones: Could user privacy profiles help? In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 201–212, New York, NY, USA, 2014. ACM.

[5] B. Liu, B. Liu, H. Jin, and R. Govindan. Efficient privilege de-escalation for ad libraries in mobile apps. In *MobiSys*, pages 89–103, 2015.

[6] G. Paul and J. Irvine. Achieving optional android permissions without operating system modifications. In *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, pages 1–5, May 2015.

[7] P. Wijesekera, A. Baokar, A. Hosseini, S. Egelman, D. Wagner, and K. Beznosov. Android permissions remystified: A field study on contextual integrity. In *Proceedings of the 24th USENIX Conference on Security Symposium*, SEC'15, pages 499–514, Berkeley, CA, USA, 2015. USENIX Association.