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 322 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

Permission Comprehension, Android, Privacy

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

Modern smartphones and tablets are powerful devices as they possess "computer-like" computing capabilities, a collection of sensors, in addition to communication via WiFi, Bluetooth, and mobile phone networks. In order to protect the user's devices from potential security risks, an app must explicitly ask the user's permission to grant it access to some of the mobile device's sensitive resources. There are two main followed approaches in asking the users to grant permissions. The first approach is to show the user all the permissions an app requests at installation time. The other approach is to let the app requests permission while the user accesses different functionalities of the app. Previous research show that displaying all permissions requested by the app at installation time is non effective in informing the users about the potential risks of installing the app [6, 13]. This mechanism fails in communicating potential risks to the users. One reason behind this risk communication problem is labeling many permissions as "dangerous permissions" even when the app has a legitimate need for such permission(s) [21]. When users install an app with "dangerous permission" and later find no negative consequences, they learn to ignore future warnings [15,23]. The second reason for the poor usability of this permission display approach is lack of adequate explanations, leaving the users confused with little understanding of what the permissions imply [7,12].

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. This approach has been addressed in numerous works since its inception which have discussed privacy and security concerns with this model, and have proposed alterations to the model to enhance the user's experience in a variety of ways. For example, the previous model was criticized for not allowing users to selectively grant the permissions their apps were requesting and would force the user to either not use the app, or trust the app to not misuse their data or device, even for the permissions that they did not feel comfortable with.

Several works proposed solutions such as custom frameworks which would provide the user the ability to accept only a subset of the app's requested permissions [4,11,16,20]. While such solutions would have provided the advantage of providing the user more control over their apps, the average Android user would be unlikely to conduct these somewhat complicated technical tasks. Additionally, these recommended changes would often require the user to root their phone, something which could lead to additional security threats [20].

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 instead of 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.

Previous works have criticized the old model of requesting permissions at the beginning of the installation process in an all or nothing fashion for a variety of reasons including the inability of a user to alter permission settings based on situational constraints such as who is using the device or their location [4, 17]. According to Android, some of their primary goals for implementing the new permission model was to streamline the app installation and update process, while providing them more control over the permissions their apps used [2].

[Pradeep: Discuss these goals in greater detail. Also, identify other concerns raised in the literature on the previous permissions model. Doing this systematically will lead to research questions of interest.]

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 [3].

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

Dangerous permissions are deemed to pose significant risks to the user's privacy, or to other apps installed on the device. One way that an app could pose a risk to another app is that through *permission leaks*, which is when a less privileged app uses the granted permission of a more privileged app through the use of inter-process message components, or *Intents*. Although the sharing of permission capabilities is an intended function of Android which serves as a way to make the user experience more friendly and not overwhelm them with permission requests [1], this can create security and privacy related issues in less secure or untrusted apps [8].
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 all permissions, including the dangerous ones, when installing an app or not install the app. The all or nothing ordeal was true when updating an app, too. That is, a user had to accept all permissions and update requires to install it or not install the update at all. 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 [24]. 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 [13] to more granular sets of Android permissions [18]. *[Pradeep: If it is not too many recommendations, let us list them all instead of just indicating the range.]* *[Dan: Is this redundant? Should we just move it?]*

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 and an 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

In the following section, we will discuss our hypothesis before conducting the study, our study design, and how our study was conducted.

3.1 Terms

Several terms that will be used in our study include:

- **Pre-M**: Android API ≤ 22, which used the old Android permission model of asking users to accept or reject all requested permissions at install time.
- **Marshmallow**: Android API ≥ 23 which allows apps to ask users to accept or reject permissions runtime.

- **M–Rationale**: The app running on Android Marshmallow, and where permission requests are given a short rationale as to why they are needed before prompting the user to accept or reject the permission at runtime.
- **M–No rationale**: The app running on Android Marshmallow, but where no extra explanation of why the permission is required is provided to the user.

3.2 Hypothesis

[work on telling a story with these][Add in hypothesis from MTurk study]

In order to guide our research, we developed several hypothesis:

1. The new permission model affords users greater control to choose the permissions that their app uses, granting some permissions an apps requests, but not others. Users may also decide to alter an app's granted permissions at any time after the installation process. A primary goal of the new permission model was to give users more control over the app's permissions [2].

   Granting permissions to an app is accompanied by risks such as downloading malicious software, and leakage of personal information. An important factor that affects risk assessment is the amount of control that users believe they have over the risky activity. Users perceive controllable activities (e.g. toggle permissions) as less risky than uncontrollable activities [10].

   H_0 = Marshmallow will not impact on how secure a user feels when using the app.

   H_1 = Marshmallow will make users feel more secure with using the apps.

2. Marshmallow adds the ability for developers to provide a greater context as to why the app is requesting its permissions. For example, developers may choose to make use of such functionality as *shouldShowRequestPermissionRationale()* to provide more information in a pop-up about why an app is requesting a permission. The role of information is vital in supporting decision making. According to the information processing theory, when people are provided with useful information, they are enabled to engage in analytical and rational thinking [5]. In the context of technological risks, users can assess the tradeoff between risk level and perceived benefit when proper information are conveyed to them [9]. Such information could be scenario information or frequency information [22].*[Dan: clean up this wording]*

   Two methods of measuring how helpful these rationales are for users is how well they help a user to understand the permission being requested, and if it has any impact on their ability to recall the permissions the app has asked for.

   H_0 = The rationale functionality that Marshmallow provides will not help a user to better understand the permissions that the app is requesting.

   H_1 = The rationale functionality in Marshmallow will make it easier for a user to understand the permissions an app is requesting.

   H_0 = The rationale provided in Marshmallow will not help users to recall the permissions the app asked for.

   H_1 = The rationale provided in Marshmallow ill impact the user's ability to recall the permissions the app asked for.

3. A basic premise behind the Android permission structure is that it is intended to inform users about the risks of installing applications [19]. Previous research has shown that users do not typically read or comprehend the permissions that their app has requested very well [7, 12]. Users can only make correct security decisions for their device if they correctly understand what the permissions mean. A system which is better able to inform users about the permissions the app was requesting would allow them to make more comfortable, appropriate choices about their privacy when using the app on their device.

   H_0 = Marshmallow will have no impact on a user's ability to correctly recall the permissions an app has asked for.

   H_1 = Marshmallow will have a statistically significant impact on a use's ability to recall the permissions their app has asked for

4. Previous research has shown that most users do not pay attention to the permissions being request by their apps

   [7]

   [cite][finish.....]

   H_0 = The new permission model of asking users to accept permissions at runtime does not impact how much a user pays attention to the permissions being requested by their app.

   H_1 = Asking users to accept or reject permissions at runtime will allow users to

   [Dan: I am not sure how to measure this... leave it in??]

4. STUDY DESIGN

In the following sections, we will describe how we conducted both our in person, and online study using MTurk.

4.1 In Person Study

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

4.1.1 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.

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

4.1.2 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 [14]. *[Dan: describe these permissions more?]* 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.

2. **Android Marshmallow - No Rationale** (API 23) 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.

4.1.3 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.

4.2 Online Study Using MTurk

Although we collected results from over **[XXX]** participants from a diverse set of backgrounds and age ranges, we still felt that there were some areas of our study to improve upon. For example, we wanted to include a second application into our study to see how our results were affected when the users were using more than one application in the analysis. We also wanted to include more variability in the rationale including alternating providing permission rationales to see any affect that would have on the user's responses. Due to the nature of our audience, we were limited in the amount of study time we had with the participants in our in-person study. In order to expand upon our results from our in person study, we decided to conduct a larger online study using Amazon Mechanical Turk (MTurk)[2], which is an online crowdsourcing environment which we used to recruit participants for our online study.

4.2.1 Recruitment

4.2.2 Applications

We used two apps for our MTurk online study, a Tic-tac-toe game similar to the one used in our in person study, and a 'Restaurant Evaluation' app. These apps were hosted on an online emulator, Appetize.io[3] which allowed study partipants to use the apps in an online environment through their web browser. We chose to use an online emulator for our study to make it was easy as possible for participants to use our apps, and to eliminate variability and mistakes that could arise from participants installing the apps on their own devices. We chose https://appetize.io because

Unfortunately, the emulator did not allow

API version used

Tic-tac-toe: The primary differences between the Tic-tac-toe game used in this MTurk study, and

Restaurant: XXX

4.2.3 User Survey

5. 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?

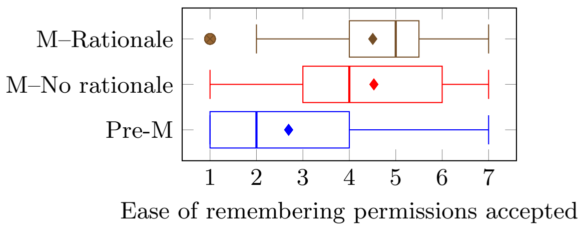
A primary goal of the new permission model was for users to be more knowledgable about the permissions their apps were using [2]. It is important for a user to recall **[Yasmine: Finish]**

As part of our user study, we asked users how easy it was to recall 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.

There was only a very small difference in the results between the 'M - With Rationale' and 'M - Without Ratio-

[2]https://www.mturk.com
[3]https://appetize.io



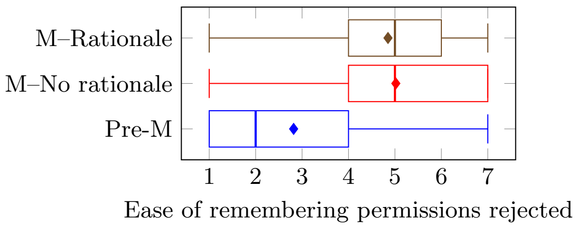


Figure 1: Ease with which participants remember the permissions they accepted and rejected.

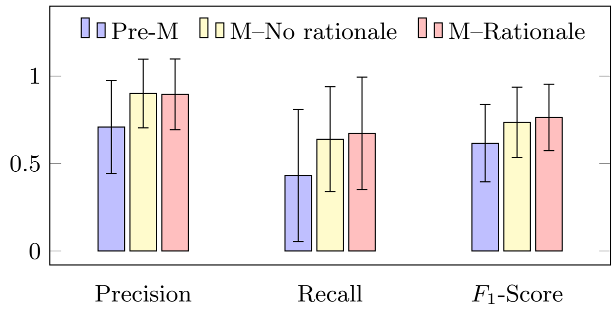


Figure 2: Participants' ability to remember permissions.

nale' 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 [7].

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: best way to say this?]* The results of this analysis are shown in Table 1.

Table 1: Permission Model User Acceptance

	Acceptance Rate	
Permission Group	**Rationale**	**No Rationale**
Location	0.95	0.86
Contacts	0.62	0.49
Phone	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?

6. OTHER OBSERVATIONS

7. LIMITATIONS & FUTURE WORK

8. 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 the Rochester Institute of Technology.

9. REFERENCES

[1] Permissions best practices. https://developer.android.com/training/permissions/best-practices.html.
[2] Requesting permissions at run time. https://developer.android.com/training/permissions/requesting.html.
[3] System permissions. https://developer.android.com/guide/topics/security/permissions.html.
[4] M. Conti, V. T. N. Nguyen, and B. Crispo. Crepe: Context-related policy enforcement for android. In *Proceedings of the 13th International Conference on*

Information Security, ISC'10, pages 331–345, Berlin, Heidelberg, 2011. Springer-Verlag.

[5] S. Epstein. Integration of the cognitive and the psychodynamic unconscious. *American psychologist*, 49(8):709, 1994.

[6] A. P. Felt, K. Greenwood, and D. Wagner. The effectiveness of application permissions. In *Proceedings of the 2nd USENIX conference on Web application development*, pages 7–7, 2011.

[7] 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.

[8] A. P. Felt, H. J. Wang, A. Moshchuk, S. Hanna, and E. Chin. Permission re-delegation: Attacks and defenses. In *USENIX Security Symposium*. USENIX Association, 2011.

[9] B. Fischhoff, P. Slovic, S. Lichtenstein, S. Read, and B. Combs. How safe is safe enough? a psychometric study of attitudes towards technological risks and benefits. *Policy sciences*, 9(2):127–152, 1978.

[10] A. R. Hale and A. I. Glendon. *Individual behaviour in the control of danger*. Elsevier Science, 1987.

[11] A. Kaur and D. Upadhyay. Pemo: Modifying application's permissions and preventing information stealing on smartphones. In *Confluence The Next Generation Information Technology Summit (Confluence), 2014 5th International Conference -*, pages 905–910, Sept 2014.

[12] P. G. Kelley, S. Consolvo, L. F. Cranor, J. Jung, N. Sadeh, and D. Wetherall. A conundrum of permissions: Installing applications on an android smartphone. In *Proceedings of the 16th International Conference on Financial Cryptography and Data Security*, FC'12, pages 68–79, Berlin, Heidelberg, 2012. Springer-Verlag.

[13] 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.

[14] 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.

[15] W. A. Magat, W. K. Viscusi, and J. Huber. Consumer processing of hazard warning information. *Journal of Risk and Uncertainty*, 1(2):201–232, 1988.

[16] M. Nauman, S. Khan, and X. Zhang. Apex: extending android permission model and enforcement with user-defined runtime constraints. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, pages 328–332. ACM, 2010.

[17] M. Nauman, S. Khan, and X. Zhang. Apex: Extending android permission model and enforcement with user-defined runtime constraints. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '10, pages 328–332, New York, NY, USA, 2010. ACM.

[18] 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.

[19] A. O. S. Project. Android security overview. https://source.android.com/security/.

[20] S. Rasthofer, S. Arzt, E. Lovat, and E. Bodden. Droidforce: Enforcing complex, data-centric, system-wide policies in android. In *Proceedings of the 2014 Ninth International Conference on Availability, Reliability and Security*, ARES '14, pages 40–49, Washington, DC, USA, 2014. IEEE Computer Society.

[21] B. P. Sarma, N. Li, C. Gates, R. Potharaju, C. Nita-Rotaru, and I. Molloy. Android permissions: a perspective combining risks and benefits. In *Proceedings of the 17th ACM symposium on Access Control Models and Technologies*, pages 13–22. ACM, 2012.

[22] P. Slovic, E. Peters, M. L. Finucane, and D. G. MacGregor. Affect, risk, and decision making. *Health psychology*, 24(4S):S35, 2005.

[23] D. W. Stewart and I. M. Martin. Intended and unintended consequences of warning messages: A review and synthesis of empirical research. *Journal of Public Policy & Marketing*, pages 1–19, 1994.

[24] 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.