Can Systems Explain Permissions Better? Understanding Users' Misperceptions under Smartphone Runtime Permission Model

Bingyu Shen[1], Lili Wei[2], Chengcheng Xiang[1], Yudong Wu[1],
Mingyao Shen[1], Yuanyuan Zhou[1], and Xinxin Jin[3]
[1]*University of California, San Diego* [2]*The Hong Kong University of Science and Technology* [3]*Whova, Inc.*

Abstract

Current smartphone operating systems enable users to manage permissions according to their personal preferences with a runtime permission model. Nonetheless, the systems provide very limited information when requesting permissions, making it difficult for users to understand permissions' capabilities and potentially induced risks.

In this paper, we first investigated to what extent current system-provided information can help users understand the scope of permissions and their potential risks. We took a mixed-methods approach by collecting real permission settings from 4,636 Android users, an interview study of 20 participants, and large-scale Internet surveys of 1559 users. Our study identified several common misunderstandings on the runtime permission model among users. We found that only a very small percentage (6.1%) of users can infer the scope of permission groups accurately from the system-provided information. This indicates that the information provided by current systems is far from sufficient.

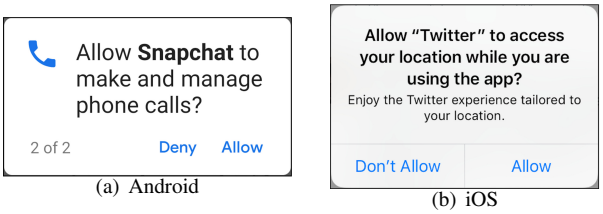
We thereby explored what extra information that systems can provide to help users make more informed permission decisions. By surveying users' common concerns on apps' permission requests, we identified five types of information (i.e., decision factors) that are helpful for users' decisions. We further studied the impact and helpfulness of the factors to users' permission decisions with both positive and negative messages. Our study shows that the *background access* factor helps most while the *grant rate* helps the least. Based on the findings, we provide suggestions for system designers to enhance future systems with more permission information.

1 Introduction

Smartphones are pervasive today [23, 52]. The latest versions of the market-dominating smartphone operating systems, Android and iOS, both provide runtime permission management to let users decide to allow or deny apps' requests to access private data, such as photos, contacts [16, 19]. However, users can make wrong permission decisions unintentionally, which may cause severe privacy leaks in the runtime permission model as shown in recent security incidents [13, 17, 18, 20]. For example, in March 2018, it was reported that Android app of Facebook collected and uploaded users' call history and SMS messages to their servers if users grant the app permissions to read these data. Users were not aware of this even though they granted the permissions themselves. Some of them surprisingly found it out after downloading and inspecting the data collected by Facebook [17].

Compared with the previous install-time model, the adoption of the runtime permission model introduces three new challenges for users to understand app permissions. First, the runtime permission model provides a shorter and briefer description for permissions requests as shown in Figure 1. Users can hardly understand what private data will be accessed from the descriptions. Second, the runtime permission model allows users to manage permissions in groups. Users need to understand the details of private data granted in each group to make informed decisions (c.f. §2.1). Third, the change from install-time model to runtime model in Android raises new security risks: old apps can bypass the runtime permission mechanism and directly obtain all the requested permissions after installation on Android 6-9.

Smartphone systems currently play a neutral and passive role in helping users understand and manage the permissions: *they only provide brief descriptions about the permission groups in permission request dialogs.* These descriptions typically contain incomplete information. Figure 1 gives two examples of permission request dialogs on Android 9.0 and iOS 13 respectively. In Figure 1 (a), the dialog informs users the permission will "allow Snapchat to make and manage phone calls". From this notice, ordinary users cannot know that the app can also collect the phone's unique ID (i.e., IMEI) once the permission is granted. In Figure 1 (b), the dialog informs users that the permission will "allow Twitter to access location". However, it does not tell whether the location information will be uploaded to the server or how it will be used. Such simple descriptions can mislead ordinary users. Our study shows that only 6.1% of users can correctly understand all the capabilities in the permission groups (cf. §5.1).



(a) Android (b) iOS

Figure 1: **Examples of permission request dialogs on Android and iOS.** In Figure 1(a), the dialog only shows that Snapchat requests a permission to make and manage phone calls; however, it does not inform users that it will also allow the app to access phone status and ID (i.e. IMEI). In Figure 1(b), the usage descriptions provided by the Twitter app only give obscure descriptions of how location data are used. App developers may have incentives not to honestly and comprehensively disclose their entire access and usage of user data [50, 57]. **From these brief descriptions, users can hardly have a comprehensive understanding of the risks of granting these permissions.**

To better explain permissions, smartphone systems give app developers the opportunity to provide explanations when requesting permissions [16, 19]. However, previous studies [50, 57] found that this has several problems. First, app developers may not provide correct explanations. Liu et al. [50] found that a significant proportion of runtime explanations state that the app only requests basic permissions yet, in fact, the app requests more permissions than the claimed ones. Second, most developers tend to only describe the benefits to the users, but hide the details on what information will be collected and how the information will be used [57]. This implies that users can be misled if the system solely relies on app developers to explain the permission requests. It is necessary to have systems provide more accurate information to help users understand and manage permissions.

Besides, it is also unclear whether systems have effectively notified their users with the risks induced by permission model changes. Android changed from the install-time permission model to the runtime model in 2015. For compatibility, Android 6.0 and later versions (referred as Android 6.0+) still support apps using SDKs prior to 6.0 (referred as low-version apps). Low-version apps can directly obtain all their requested permissions after installation. Our study shows that 38.3% of our 180 surveyed Android 6.0+ users mistakenly expect low-version apps to request permission at runtime (cf. §4).

While many previous works focused on user comprehension of the *install-time* permission notices [37, 39, 43], it is still unclear how well users can understand the permissions in the *runtime* model. In this paper, we aim to identify the problems in the runtime permission model and evaluate their impacts on users by answering the following research questions:

RQ1. (Risks induced by permission model compatibility) How commonly do users have low-version apps installed, which may take advantage of permission model compatibility to bypass runtime user consents?

RQ2. (Runtime permission comprehension and management) Can the information provided by the system help ordinary users to precisely comprehend the permissions and their capabilities? How often do users review their permission settings after they granted them at apps' runtime?

RQ3. (Extra information from the system) What extra information (if the system can provide) would impact users' permission decisions?

To answer the research questions, we conducted three different types of studies. We first collected app permission settings from 4,636 real mobile users to study the real-world adoption of the runtime permission model. We then conducted both the interview study (n=20) and two online surveys (n= 359 and n=1200) to study users' comprehension and experience with permissions. We also identified factors that are of users' concern in making permission decisions in our interview study and used online surveys to investigate their impact on users' decisions.

Our study reveals three interesting findings: (1) Low-version apps are still widely used three years after the runtime permission model was introduced. Among the 4,636 studied Android users, 61.8% have at least one such app installed on their devices (§4); (2) Only 6.1% of survey respondents can accurately infer the scope of all the permission groups after they read permission explanations from the smartphone OSes (§5); (3) Messages capturing negative aspects of the apps are more likely to impact users' permission decisions (§6).

This paper makes the following three major contributions:

- We study users' understanding of the information provided by smartphone systems for the runtime permission model. We identify common misunderstandings raised by the permission model change, the design of permission groups and app-provided explanations.
- We identify five factors that users are concerned about in making permission decisions and quantitatively compared their impact on users' permission decisions from both positive and negative perspectives.
- Based on our study findings, we provide recommendations to the designers of smartphone OSes to address common misunderstandings of the runtime permission model.

2 Background

2.1 Permissions & Permission Groups

Permissions are introduced to gain explicit consent from smartphone users to access sensitive data or system resources. Smartphone OSes organize permissions as permission groups and let users decide whether to allow or deny each permission group [15]. For example, in Android, the `READ_SMS` and `RECEIVE_SMS` permissions are included in the `SMS` group as a whole. When either `READ_SMS` or `RECEIVE_SMS` is requested, Android will ask users for `SMS` group with the same notice.

Users have two ways to manage permissions in the runtime permission model. First, users can make permission decisions in dialogs when they are using an app, as shown in Figure 1(a). However, these system dialogs are not informative due to: 1) they display the same message when any permission in a permission group is requested (e.g. in Figure 1(a), the message could be displayed when the app requests to access phone ID or make phone calls); 2) they only give brief permission explanations, which is not intuitively understandable (e.g. in Figure 1(a), "make and manage phone calls" also includes accessing phone status and ID). Second, users can grant or revoke an app's permissions in privacy settings afterwards. However, the system settings provide neither detailed explanations nor clear definitions of permission groups, as shown in Figure 2. Users may need to guess the relevant resources allowed by each permission group from the group name.

2.2 Permission Management

On Android. Since version 6.0 which was released in 2015, Android has changed its permission model from *install-time* to *runtime* permission model for four years at the time of study in 2019. In the install-time model, a list of requested permissions and their descriptions are shown before the installation of an app, as shown in Figure 3. Users can either (1) grant *all* the requested permissions to the app or (2) reject and terminate the app installation. In the runtime permission model, permission request dialogs are shown to requested permissions when users start using the app, so that users can make decisions at the granularity of the permission group.

Compatibility becomes a problem for phones which support runtime permission model. Android developers need to set a target SDK version in the configuration file to specify the Android version that the developers have tested against. Apps with a lower version of SDK cannot accommodate to runtime permission model: all requested permissions are granted at installation time even when running on newer versions of Android. If users are unaware of this issue, they may expect that all apps will request permissions at runtime and unintentionally grant all permissions at install time.

On iOS. iOS has been using the runtime permission model since iOS 6 in 2012 [12]. It has two key differences compared to Android. First, iOS has a finer-grained permission model for some sensitive information. For example, iOS users can independently manage permissions for specific categories of personal data like step count or heart rate within the `Health` permission group, as well as control read and write permission for each category. We took the different permission groups into account when we design the study for iOS and Android.

Another key difference is that iOS *requires* app developers to specify a usage explanation for each of the requested sensitive resources as shown in Figure 1(b), which is optional and recommended on Android [16]. However, developers may provide partial or misleading explanations that avoid users denying their apps' permission requests [50, 57].

3 Methodology

3.1 Permission System Evolution Study

To study potential issues of permission model change, we designed and implemented an Android app to collect apps installed on users' phones and their permission settings from *real* users. We then used the collected data to analyze the evolution of apps' target versions on the Android market and users' permission decisions for various apps. We did not gather data from iOS due to: (1) iOS does not allow a third-party app to get other apps' permission granting status. (2) Users are not able to install apps with the install-time permission model since iOS 6 in 2012.

Compared to simply crawling apps and relevant data from app markets, the real permission settings enable us to know (1) users' actual permission settings (allow or deny) for the apps and (2) the impact of low target SDK version apps, such as the percentage of users who have installed such apps.

Data Collection Methodology. We designed an Android app, Permission Checker (PerChecker), to help users see the list of detailed permissions of each app under permission groups [9, 34]. The app was released on Google Play in June 2018 and has received over 10k downloads by June 2019. For each user, we leveraged PerChecker to collect the list of installed apps, as well as each app's requested and granted permissions. We also collected the IP and MAC address as the unique ID for analytical purposes. No personal demographics data was collected from PerChecker users. Our data collection process was from June 2018 to September 2018. To boost the initial installs, we used Google Ads in August 2018 with keywords "permission" to get the first 400 installs, then stopped advertisements.

To keep the data collection process transparent to users, we provided a clear summary of privacy policy [8] clarifying what data will be collected and how the data will be used, which will be shown on the app's first launch. Users can also opt out of the data collection at any time. In total, we collected 4,636 permission settings from distinct Android users whose phones support the runtime permission model. The dataset is available at [7].

3.2 Interview Study

We conducted semi-structured interviews to study users' comprehension of permission groups and related risks, as well as factors that affect users' permission decisions. The participants must have smartphones of the runtime permission model. The interview results are used to design surveys in §3.3. Before the interview, we refined the questions through a pilot study with people from various knowledge backgrounds and verified the questions' intelligibility. The full interview questions are available online [10].

Interview Design Methodology. Our semi-structured interviews are guided with predefined questions. We also encouraged the participants to talk about their understandings

of any related topics. Here are the main interview phases (for full phases and questions, please refer to [10]).

(1) Permission group comprehension. We asked if they can find apps' permission settings on a smartphone. If users failed to find the permission settings, we would help them find the settings. Then we presented the participants with a list of permission groups in settings as shown in Figure 2 (We also use the centralized settings display for iOS). Then we randomly picked 4-5 permission groups and asked participants to explain what resources each permission group controls. We further asked how often the participants used the permission settings and whether they would check them regularly.

(2) Permission model changes. (Android only) We showed the participants the prompt when downloading the "Camera FV-5 Lite" (an app with low target SDK version) with the provided phone, and asked if the permissions will be granted immediately after click "Accept" and whether there will be permission dialogs after they start using the apps, and reasons.

(3) Permission rationale. (iOS only) We showed the screenshot of `Camera` permission request for "Prisma" (an app to stylize photos, see Figure 7) on iPhone, and asked whether the rationale in the dialog is from the systems or the app developers. We also asked for reasons and whether they find this message helpful.

(4) Concerns in granting permissions We asked if they have met uncomfortable permission requests and also asked for specific details such as the permission and app's name. We further asked why they found the permission requests uncomfortable or their concerns when they were making permission decisions. Then we asked what factors they would consider in making permission decisions and required them to provide some examples to improve the reliability of results.

Recruitment. We spread the advertisement on the bulletin board in public places like malls and parks. We advertised our study as "Behavior observation with smartphones" study without mentioning privacy or security to reduce the recruitment bias for people who are in favor of privacy or security questions. Before the interview process, we first confirmed the participant is qualified for the interview: the participants must own an iOS device or an Android device with version 6.0 or higher. 20 participants satisfied the interview requirements and fully completed the interview. The demographics are reported in §3.4. Our interview was conducted in a coffee shop (a casual environment) and was recorded for future notes-taking. Each interview took 10–30 minutes (avg. 14.3 mins). Each participant was compensated with a $5 gift card.

Data Analysis Procedure. We gathered the text by transcriptions, and then we performed an analysis on the data with three steps. First, we read through the data and divided the data into sections based on the interview questions. Second, one researcher reviewed the text to obtain the initial codebook from each of the sections. Different coding methods are used for data from different interview phases. For the first three phases, deductive coding was used because the codes are mostly expected. We added new codes if we found any and reorganized the codebook as we code on. For the last phase (i.e., *concerns in granting permissions*), inductive coding was used for the initial codes, which follows the coding practices for exploratory data in social science studies [31]. The codebooks are gradually refined through multiple readings and interpretations. We identified five subthemes under the *decision factor* theme based on the codes. Third, we have two other researchers independently review the data to assess the coding reliability. Two researchers met and discussed the cases where their codes differed, and converged on all final codes. The coding reliability is measured with the Krippendorff's α statistic [42] and the result is shown in Table 10, indicative of largely consistent coding. The codebook with description and examples are presented in Table 10 and Table 11.

3.3 Internet Survey

3.3.1 Survey Structure

We conducted *two* separate surveys to study users' comprehension of the permissions (*Permission Comprehension*) and the factors of their concern when making privacy decisions (*Decision Factors*), respectively (see [10] for survey instruments). Both surveys contain a background section including demographics questions to screen and filter the responses.

Survey 1: Permission Comprehension.

Permission Group Comprehension. In the runtime permission model, permissions are managed in permission groups. This survey aims to investigate if users can understand the scope of each permission group (i.e., their protected resources and permitted actions) based on the systems' descriptions.

To achieve this, in each question of this section, we first presented a permission request dialog (Figure 1) to our respondents and then asked them to choose what actions can be performed by the app once the permission is granted. We provided our respondents with the shuffled correct choices describing relevant actions controlled by the permission group and two more incorrect choices describing irrelevant actions. We also provided choices, "None of these" and "I don't know" in case our respondents find no choices applicable in their understanding or they were not sure about the answer. Our questions in this section covered all the permission groups shown in Table 1. To avoid overwhelming the respondents with too many questions, only four questions for different permission groups are randomly drawn for each respondent.

Permission Model Changes (Android Only). As discussed in §2.2, the evolution of the permission model on Android makes it possible for apps to get the permissions granted from users unintentionally. In this section, we investigated whether our Android respondents were aware of such permission model changes and their induced risks. The permission notice for low-version apps in the app market is shown before downloading apps, as shown in Figure 3. We then asked our respondents three questions: (1) Will the app be able to access

Table 1: Table of permissions that require user consent in runtime on the recent version of Android and iOS. Y means the app needs prompt users to get this permission; X means the app has no way to get this resource on this platform; - means the app does not need to prompt users for this permission or does not need to have such permission. We study all permissions marked with Y in this paper.

Permission(s)	Android 9.0	iOS 12
Calendar, Camera, Contacts, Microphone	Y	Y
Location	Y	Y[1]
Body Sensor	Y	Y[1]
Storage	Y	X
Photos	Y[1]	Y
SMS, Call Log, Phone	Y	X
Bluetooth Sharing	-	Y
Music & Media, Health	-[1]	Y

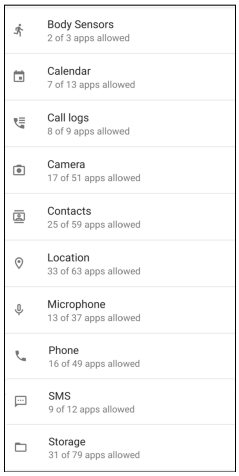


Figure 2: Android settings for app permission groups. Users need to infer the scope of the group to manage permissions.

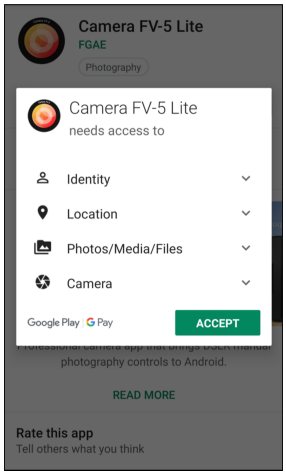


Figure 3: Google Play's notice for apps with low target SDK version before installation. It gives no clear warning to users.

the listed resources immediately after clicking accept? (2) Can you change the permission settings again? (3) Will the app ask for the permissions again after launching the app?

Survey 2: Decision Factors. This survey focuses on factors that can influence users' decisions on permission requests (*decision factors* for short). Specifically, we studied *six* decision factors: *reviews, rating, brand reputation, background access*, *data transmission* and *grant rate*. The detailed definitions of each decision factor is presented in §6.1. Our study compares how users' permission decisions are impacted by positive and negative messages of each decision factor. Our study also surveys users' opinions on how helpful they find different factors in making permission decisions.

[1] *Location* permission decisions on iOS are divided into 1) always allow, 2) allow only while using this app, 3) deny; *Body sensor* permission on iOS is called Motion; *Photos* on Android is regarded as common external storage, so it belongs to Storage permission; *Music & media* library are specific to Apple services.

In our survey, each respondent is provided with three simulated scenarios of permission request for three different permission groups (`Contact`, `Calendar`, and `Location`). These permission groups are selected as they are available on both Andriod and iOS.The scenarios are as follows:

(1) *Felp requests Contact permission.* Felp can help you find good restaurants around you. In Felp, you can read the menu and other users' reviews for restaurants.

(2) *RShare requests Calendar permission.* RShare can help you find a ride and carpools. You can search for your destinations and find suitable rides in RShare.

(3) *LCGE requests Location permission.* LC Gas & Energy (LCGE) is an app to help you pay your electronic bills and utility fees. You can view your current energy usage and fees, make payments or schedule a payment.

We used these descriptions to introduce app functionalities to our respondents. In order to provide more runtime context, we also provided screenshots of the apps before and after the permission requests, with the triggering actions, as shown in Figure 4. The images are adapted from screenshots of real apps to simulate request context and help users understand the scenarios. We decided to use action-triggered permission requests to represent the runtime context, because most apps will request the three permissions interactively via clicks [51].

The respondents need to decide to allow or deny the requests for two times: (1) the first time is as implemented in the current smartphone systems without any additional messages and (2) the second time is with additional messages regarding one of our six decision factors under study. We aim to investigate whether the respondents can make different permission decisions with and without these additional messages. We provided each respondent with either positive or negative messages regarding only one factor to avoid the influences of different factors on their perception.

The messages for each factor are shown in Table 2. We chose the messages for each factor based on the interview study results and refined based on our pilot study. We framed the messages both from positive and negative side based on the characteristics of each decision factor. For the factors presented with specific values (i.e., rating, review and grant rate), we chose the values at extreme in order to clearly present the positive or negative side of the factors. For *brand reputation*, to have a unified standard to present the app's reputation in protecting users' privacy (c.f. §6.1), we used whether the app has been certified by two recognized standards (GDPR [3] and ISO/IEC 27001 [5]). To help users understand these two standards, we provided brief information about these certifications and standards in the question text.

After answering the three simulated scenario questions, respondents can better understand the studied factors. We then asked the respondents to rate the helpfulness of the decision factor in a five-point Likert scale. The full questionnaire with all scenarios, factors and their related messages are in [10].

Permission comprehension. In order to study the correlation

Table 2: Factors and their messages displayed in the permission dialogs. `App` is the simulated app name and `Resource` is the permission group name that an app requests. The text in the brackets will be displayed for positive messages.

Factors	Messages
Background access	`Resource` will [not] be accessed when you're not using the app.
Data transmission	`Resource` will [not] be transmitted and [or] stored by `App`.
Rating	The rating of `App` is 2.1 [4.8] rating in app store.
Review[1]	`App` has 13 [no] reviews related to `Resource` in app store.
Grant rate	10% [90%] of `App` users granted `Resource` access.
Brand reputation[2]	`App` has [not] been GDPR certified and [or] ISO/IEC 27001 certified.

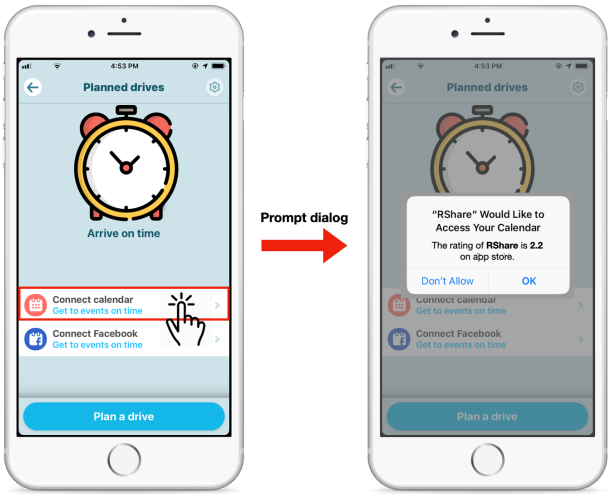


Figure 4: **The UI transition figure[3] for simulated scenario RShare in survey 2**. The respondent was first shown figures without the message in the prompt dialog. The positive and negative messages for all factors are in Table 2.

between users' comprehension and their permission decisions, we included the permission group comprehension questions as described in survey 1, for permission groups used in the scenarios (i.e., `Contact`, `Calendar`, and `Location`).

Background Section. Both surveys contain a background section, including the OS version information and demographic questions. (1) *OS Version.* We required our respondents to provide their OS versions, because the correct answers to the questions in *Permission Group Comprehension* may vary for different OS versions. Besides, respondents whose Android phone' OS versions are below Android 6.0 will exit the survey immediately. (2) *Demographics.* We collect demographic data of respondents including their gender, age, education level, experience in computer science or related fields, privacy knowledge level and occupation.

[1]For the negative framing for *review* factor, we display example negative reviews in the question descriptions. One example is "Why `App` needs my `Resource`? Intrusive and unnecessary permission invasion of privacy!".

[2]Since users may not be familiar with GDPR and ISO/IEC 27001, we include the short descriptions in the question text. GDPR certification means the company is transparent and honest in collecting and protecting users' personal data. Appropriate technical and organizational measures have been taken to achieve data protection; ISO/IEC 27001 certification means that the company has defined and put in place best-practice information security processes to prevent security risks such as hacks, data leaks or theft.

[3]Some icons are from Freepik.

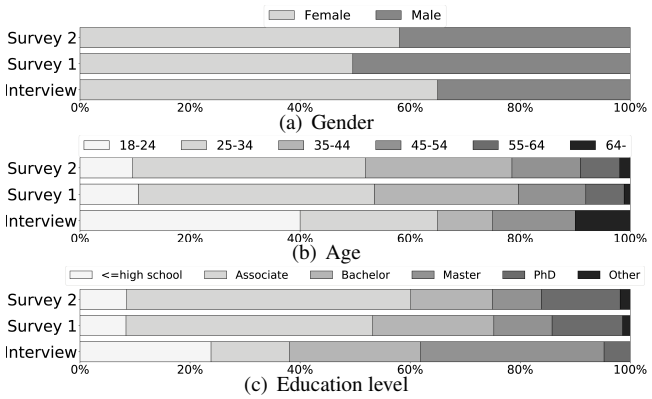


Figure 5: Key demographics for interview study (n=20), Survey 1 (n=359) and Survey 2 (n=1200).

3.3.2 Recruitment

We conducted our surveys on Amazon Mechanical Turk (AMT) from April 2019 to May 2020. We required that AMT workers must be at least 18 years old, have a 98%+ approval rate in their task history. To avoid bias in recruitment, we avoided using terms related to security and privacy in the AMT task description. We required respondents to have a smartphone when answering questions.

We received 359 valid responses for survey 1 (180 for Android and 179 for iOS) and 1200 valid responses for survey 2 (600 for Android and iOS each). The average time spent on survey 1 and survey 2 are 6.78 and 7.81 minutes. Each respondent was compensated $1 for completing the survey.[4]

3.4 Demographics

The key demographics for interview study (n=20), survey 1 (n=359) and survey 2 (n=1200) are shown in Figure 5. In the interview, survey 1 and survey 2, 16 (80%), 279 (77.7%) and 843 (58.0%) of respondents reported no experience (education or job) in computer science, IT, or related field respectively.

3.5 Ethical Considerations

Our PerChecker study was conducted under a collaboration between a company and a university. The PerChecker app was developed and released by researchers from the company. We took various measures to ensure that users' privacy is respected. First, the researchers were trained with ethics for user research before the study. Second, before collecting any data,

[4]We increased the compensation rate for survey 1 from $0.5 to $1 after we found that the time spent on survey 1 was longer than that from expected. The compensation change was approved by IRB.

PerChecker explicitly requests for users' approval for data collection and usages. Only after users approve the request, PerChecker starts collecting data. Third, during the usage of PerChecker, users can choose to delete all data collected before and disallow future data collection. Fourth, all data transmissions to our server are encrypted via AES256. Fifth, no personal identifiable information (PII) is saved in the study. To differentiate data from different devices, we save the hash value of MAC and IP addresses but do not save the original addresses. The researchers at the university conducted the analysis based on the collected data. The university's Institutional Review Board (IRB) was contacted and concluded that the study did not require IRB review.

The interview and Internet survey study was conducted in a university. Before the study, we contacted the university's IRB and received an exempt for IRB review. Both studies record no personal identifiable information and thus are anonymized.

3.6 Threats to Validity

Like other user studies, our study also has a risk that the findings may be biased to the studied users and not representative enough of the entire population. In this paper, we use multiple data sources to cross-validate our findings. Our data sources include (1) real users' permission settings from PerChecker (n=4636), (2) user study with both interviewees (n=20) and online AMT workers (n=1559), each covering a large number of real mobile users. We discuss the potential limitations of each data sources below:

PerChecker. The demographics for PerChecker users were not obtained. PerChecker is listed in the "Security Apps" category in Google Play. Users find our app either by searching or recommendations. These users can be more security-sensitive since they use security tools, or less tech-savvy because they turn to tools for help to deal with permissions. Therefore, we urge the readers to take the potentially unbalanced demographics in mind when interpreting the PerChecker's statistics.

User study. A general limitation of user study is that results are self-reported and the behavior and perceptions from participants may differ from real-life conditions [46]. The participants may answer questions based on what they desire to do, not what they actually did [45], even though the previous study also shows that the findings are still correlated with real-life experiences strongly [55]. We took several measures to mitigate the limitations: in the open-ended questions, we reminded participants that there were no standard answers and they were encouraged to discuss based on their past experiences. We also mimicked the real screen interface and described triggering actions to provide runtime context in simulated scenarios in survey 1 and 2 [10].

4 Permission Model Change

Android starts to support requesting permission at runtime since Android 6.0. For compatibility, the new version systems still support early-developed apps adopting the install-time permission model (i.e., targeting versions before Android 6.0, *low-version apps* for short). All permissions requested by such low-version apps will be granted immediately after installation without asking for users' consent at runtime, even on new version systems. We investigate the significance of the problem by analyzing (1) the prevalence of installed low-version apps among real Android users and (2) users' comprehension of the system's notice for low-version apps on new Android versions with runtime permission capabilities.

Finding 1: *Three years after the introduction of the runtime permission model, low-version apps are still prevalent. Among the 4,636 real Android users in our study, a large percentage (61.8%) have at least one such app installed.*

More than half (61.8%) of PerChecker users have at least one low-version app installed (95%CI $[60.4\%, 63.2\%]$). One-in-fifteen (6.7%) of PerChecker users have five or more low-version apps installed (95%CI $[6.1\%, 7.3\%]$). Some examples of these low-version apps are shown in Table 4. We further analyzed the top 40 most-used low-version apps, and made the following observations.

The majority of these most-used low-version apps available on market are still actively updated without adding support for the runtime permission model. 70% of our analyzed low-version apps are still actively updated within 3 months when we conducted the study (September 2018). Only a small percentage (5%) are actually "legacy" apps that did not provide updates for more than two years. The breakdown in Table 3 shows that the app developers, even when the systems provide an option to better protect users' security and privacy, may not choose to update. The developers may either (1) lack the motivation or incentive to update the app with a secure higher API version, or (2) intentionally take advantage of the design flaws to collect users' sensitive data without users' attention. The second reason may become more possible when the app still keeps updating after the permission model change.

Table 3: Last update time on Google Play of top 40 common low-version apps in PerChecker users as of September 2018.

Last update	3 months	6 months	12 months	24 months
# (%)	28 (70%)	34 (85%)	37 (92.5%)	38 (95%)

Table 4: Top 5 commonly-used low-version apps among PerChecker app users. The app's metadata information is retrieved from Google Play on Sep. 20, 2018.

App Name	**Category**	**# install**	**Update date**
TextNow	Communication	10M+	19/09/18
ES File Explorer	Productivity	100M+	17/09/18
Settings DB Editor	Tools	100K+	01/09/18
WiFiman	Tools	100K+	30/08/18
Advanced Tools	Tools	100K+	27/07/18

Low-version apps have large user bases. Table 5 lists the number of downloads of each low-version app on Google Play. Note that we only analyzed the downloads from the

Table 5: Download times on Google Play of top 40 commonly-used low-version apps among PerChecker users.

Downloads	100M+	10M+	1M+	100K+
# (%)	5 (12.5%)	18 (45%)	25 (62.5%)	39 (97.5%)

biggest app market, Google Play. The actual number of total downloads for the low-version apps can be even larger if some third-party markets are included. If low-version apps exhibit malicious behavior by abusing some of the permissions, a large number of users can be potentially affected, since all the permissions are granted immediately after installation.

Finding 2: *More than one-third (38.3%) of users are not aware of the behavioral differences in requesting permissions between low-version apps and apps supporting the runtime permission model.*

In dealing with the low-version apps, Google Play only notifies users of the permissions requested by the app before its installation with a dialog (Figure 3). In the interviews, we showed the participants an example dialog when installing the app "Camera FV-5 Lite" from Google Play and explored if they could infer the different behaviors of low-version apps. *Seven* out of the ten Android participants mistakenly believed that the app would prompt permission requests again after installation - just like apps with higher target SDK versions, which shows that some users misunderstand that permissions will be not granted upon installation for the low-version apps.

In survey 1, we validated this observation in a larger population. More than one-third (38.3%) Android respondents mistakenly believed that the app would ask again for the permissions. This means that many users cannot infer the difference between low-version apps and other apps supporting runtime permission from Google Play's notification dialog. For permission management for low-version apps, the majority of respondents (80.0%) correctly knew that they could revoke granted permissions. However, among all PerChecker users with low-version apps installed, only *one* of them actually revoked permissions for a low-version app, which suggests the ability to revoke permissions for low-version apps have not been well utilized by the users in real-world scenarios.

Discussion on recent app market policy change. While we were conducting our study, Google Play began to limit the target version of newly uploaded apps [11]: staring from Nov. 2019, new apps and app updates need to target Android 9 or above. This both confirmed and mitigated the issue of low-version apps. But the issue has not been completely eliminated even with this limit. First, the policy only applies to the newly uploaded apps; for a large number of existing apps, they still use low-version SDK. Second, low-version apps can still exist on manufacturer app stores (e.g. Google Play) and third-party app stores (e.g. F-Droid) [6].

For low-version apps, Android asks users to decide to revoke dangerous permissions or not when the app launches for the first time since Android 10 [22]. However, this cannot completely eliminate the issue. First, making decisions before using the app still lacks the runtime context even in the runtime permission model. Second, the adoption of Android 10 may take a long time due to the problem of Android fragmentation [1]. These users may still be impacted by apps with low-version SDK if their OSes are not updated.

Answer to RQ1: Low-version apps are still prevalent three years after the introduction of the runtime permission model. Besides, many users mistakenly believe that the low-version apps still need to request permissions at runtime.

5 Runtime Permission Comprehension

In this section, we investigate (1) whether users can precisely infer the scope of permission groups from the system-provided messages in permission dialogs, and (2) how often users review their permission settings and whether reviewing permission settings can help them. As for iOS, the permission request dialogs mix system-defined permission descriptions with explanations provided by apps. We also investigated how iOS users perceive app-provided descriptions in the dialogs.

5.1 Permission Groups Comprehension

In our survey 1 on permission comprehension, respondents need to answer four questions related to the scope of permission groups as described in §3.3. The details of survey 1 results are shown in Table 6.

Finding 3: *Only a small percentage (6.1%) of survey respondents can correctly infer the accurate scope of permission groups from messages in the system dialogs. Users can both (1) mistakenly include choices seemingly correlated to a permission group, and (2) exclude correct choices which are hard to infer from the system descriptions.*

Around one-in-twenty (6.1%) respondents can select all correct usages of the permission groups in the comprehension test. Other respondents can be divided into two categories. One category of users (34.4% for Android and 21.2% for iOS) can respond correctly but underestimate the scope of permission groups when there are multiple correct choices, as shown in Table 7. The other category (more than 60% for both Android and iOS) overestimates the scope and includes wrong choices in their answers.

We further analyzed the most common wrong choices that are made by more than 20% of respondents in the comprehension questions. One observation is that the descriptions and names of the permission groups mislead users and cause them to select seemingly-correlated wrong choices. For example, around one-in-three (37.3%) of users believe after granting the `Camera` permission, apps can also read the pictures and videos, which is controlled by permission group `Storage` on Android or `Photos` on iOS. On Android, two-in-five (38.5%) of respondents think that apps can read contacts with the `Call Log` permission group, which is controlled by another permission group [21]. These misunderstood permission groups are designed to serve relevant functionalities, but the explanations

Table 6: Permission group comprehension results. ✓and ✗mark correct and incorrect responses respectively. - means the option is not provided for the OS platform. The options were shuffled and only four questions were randomly drawn for one respondent. "None of these", "I don't know" and other wrong options were omitted if the numbers are less than 5 (7%) for both platforms. The Music and Media, and Bluetooth permission groups on iOS are also omitted here. Full results available at [10].

Permission Group	Options	Android		iOS	
Calendar	✓Save events to your calendar	44	62.9%	63	84.0%
Android: Allow [App]to access your calendar	✓Read your calendar	61	87.1%	69	92.0%
iOS: [App] would like to access your calendar	✗ Make phone calls	9	12.9%	0	0%
Contact	✓Read your contacts	65	90.3%	62	88.6%
Android: Allow [App] to access your	✓Save new contact to your phone	24	**33.3%**	29	**41.4%**
contacts?	✓Read your Google account email address	17	**23.6%**		-
iOS: [App] would like to access your	✗ Read your location	6	8.3%	8	11.4%
contacts.	✗ Make phone calls	9	12.5%	6	8.3%
Camera	✓Take pictures and record videos	62	88.6%	65	90.3%
Android: Allow [App] to take pictures and	✗ Read pictures and videos	29	**41.4%**	24	**33.3%**
record videos?	✗ Read your contact	3	4.3%	9	12.5%
iOS: [App] would like to access the camera.	✗ Read your location	8	11.4%	6	8.3%
Microphone	✓Record your voice	55	88.7%	72	91.1%
Android: Allow [App] to record audio?	✓Record your voice when the app is in the background (b.g. for short)	39	62.9%	42	53.2%
iOS:[App] would like to access the mic.	✗ Make a phone call		-	16	**20.2%**
Location	✓Read your location	74	90.0%		-
Android: Allow [App] to access this device's	✓Read your location when you're using the app		-	53	82.3%
location ?	✓Read your location when the app is in the b.g.		-	53	82.3%
iOS: [App] would like to access your	✗ Make phone calls	11	13.4%	6	9.4%
location. (*Always allow* is chosen)	✗ Read your photos	8	9.8%	7	10.9%
Body Sensor	✓Read your steps count	46	63.0%	48	78.7%
Android: Allow [App] to access sensor data	✓Read your heart rate history	53	72.6%		-
about your vital signs?	✗ Read your fingerprints	22	**30.1%**		-
iOS: [App] would like to access your motion	✗ Read your face ID	11	15.1%		-
& fitness activity.	✓Read the info. from sensors on your phone		-	45	73.8%
	✓Read your running history		-	45	73.4%
	I don't know	12	**16.4%**	4	6.6%
Phone (Android only)	✓Get your phone number	32	**47.0%**		
Msg: Allow [App] to make and manage	✓Get your phone unique ID (e.g. IMEI)	16	**23.5%**		
phone calls?	✓Make phone call	54	79.4%		
	✓Answer phone call	45	66.2%		-
	✓Know whether the phone is making phone calls	42	61.8%		
	♢ Read call history	36	52.9%		
	✗ Read your location	9	13.2%		
Storage (Android only)	✓Read this app's photos, media, and files	52	78.8%		
Msg: Allow [App] to access photos, media,	✓Read other app's photos, media, and files	33	54.5%		
and files on your device?	✓Save new photos, media, and files	36	50.0%		-
SMS (Android only)	✓Read your SMS messages	59	79.7%		
Msg: Allow [App] to send and view SMS	✓Send SMS messages	64	84.6%		
messages?	✗ Read your location	11	14.9%		-
	✗ Make phone calls	7	9.5%		
Call Log (Android only)	✓Read your call history	74	89.1%		
Msg: Allow [App] to access your phone	✓Save new call record	31	**37.3%**		
call logs?	✗ Read your contacts	32	**38.5%**		-
	✗ Get your phone number	25	**30.1%**		
Photo (iOS only)	✓Read all photos on the device			66	94.3%
Msg: [App] would like to access your	♢ Delete photos on the device		-	13	18.6%
photos.	✗ Read all files on the device			7	10.0%
Health (iOS only)	✓Read your steps count			48	70.6%
Msg: [App] would like to access and update	✓Store your steps count		-	55	80.9%
your health data in Steps. (A separate page	✗ Read your heart rate			33	**48.5%**
with requested health data will be shown)	✗ Read your workouts history			31	**45.6%**

♢ *Read call history* moved to a new `Call Log` group since Android 9.0 [21]; *Delete photos on the device* will need extra confirmation.

Table 7: Respondent categories breakdown based on the comprehension question results. (Android $n = 180$, iOS $n = 179$). Based on the respondents' answers, we classified respondents into four categories: *All Correct* if all answers of the respondent are correct; *Partially Correct* means at least one of the respondent's answers is partially correct and no answers are wrong; *All Wrong* if all the respondent's answers are wrong; *Wrong* otherwise.

Category	Android		iOS	
All Correct	7	(3.9%)	15	(8.4%)
Partially Correct	62	(34.4%)	38	(21.2%)
Wrong	96	(53.3%)	119	(66.5%)
All Wrong	15	(8.3%)	7	(3.9%)

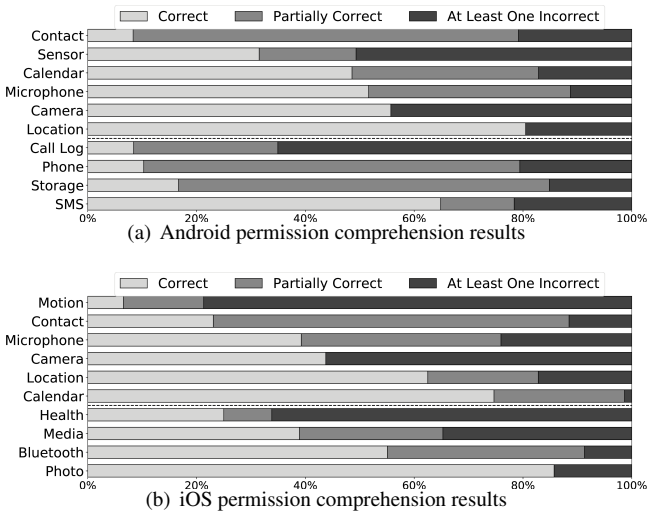
Table 8: Permission group granularity and corresponding correct response rates on Android. The correct response rate tend to be higher for permission groups with smaller number of permissions.

# of permissions	Permission group	Correct response (%)
1	Camera	55.7%
	Location	80.5%
2	Call Logs	8.3%
	Sensor	31.5%
	Calendar	48.6%
	Microphone	51.6%
	SMS	64.9%
3	Contact	8.3%
	Storage	16.7%
5+	Phone	10.3%

are not clear enough to help users understand and differentiate the actual capabilities of permission groups.

Another characteristic of wrong choices is that they are related to some critical resources that are hard to infer simply from the system descriptions. The system permission dialogs provide the most direct notices to users when users make permission decisions, but they only provide partial information on what is given away after granting permission. Take `Phone` permission group as an example, it protects phone-related features such as making phone calls and accessing unique IDs of the phone (e.g., IMEI number), but the system message is only "make and manage phone calls". In the survey results, around three quarters (76.5%) of our respondents do not know that the app can access IMEI after granting the permission, which can be used to track the app users (Table 6).

In the `Location` question for iOS users, the permission dialog contains a button, "Allow only while in use", which was introduced in iOS 11 in 2017. With this information, the majority (83%) of iOS respondents correctly understood that their location may be accessed when the app is running in the background if they selected "Always allow". This is different from the previous finding that only as few as 17% of the users knew that the background applications may have the same capability as the foreground applications in 2013 when the dialog does not contain such information [58]. This suggests that through proper notices from systems, users can better comprehend the capability of the permissions.



(a) Android permission comprehension results

(b) iOS permission comprehension results

Figure 6: Answer category breakdown for permission comprehension questions on Android and iOS. Permissions groups above the dotted line are shared by Android and iOS even though minor differences as shown in Table 1. The others are unique to the platforms. Rows are sorted based on the percentage of *Correct* answers. *Correct* means that all of the correct choice(s) for the question are selected; *Partially correct* means that not all correct choices are selected and no wrong choices are selected; *At least one incorrect* means that one or more wrong choices are selected.

Finding 4: *Users are more likely to misunderstand "coarser-grained" permission groups that control more permissions, sensitive resources, or their associated actions.*

Figure 6 shows the distribution of different answer categories (*All correct*, *Partially correct* and *At least one incorrect*) for each permission group. Different permission groups have different percentage of correct answers. The percentage of *All correct* answers for `Contact`, `Phone`, and `Call Log` permission groups on Android are lower than 10%. On the contrary, most (80.5%) answers for `Location` are correct.

To understand why users have poor understandings of certain permission groups, we investigated the relationship between the granularity of permission groups and common misunderstandings on certain permissions. The granularity of the permission group in the runtime model refers to the number of similar capabilities grouped by the system. Note that only Android defined specific permissions under permission groups, so we only study Android. We divided the permission groups according to the number of permissions in them. Table 8 shows each permission group and the percentage of completely correct answers. We find that the average correct percentage has a negative correlation with the number of permissions within the group on Android (Pearson coefficient r=-0.885, p=0.114; two-tailed). One outlier is the `Call Log` permission group with 2 permissions but has a low correct percentage. This group has many related functions related to phone calls which may cause confusion without clear explanations as described in Finding 3.

Table 9: Respondents' *initial grant rate* comparison based on the comprehension question requests in survey 2. (n=600 for iOS and Android). The initial grant rate is the percentage of respondents who choose "Allow" before we show them with the messages of decision factors. *Correct* refers to the percentage of respondents who allowed the permission request *and* correctly comprehended the permission in this scenario, while *Incorrect* means the percentage of respondents who allowed the permission request *but* incorrectly comprehended it. *p* value is calculated based on Mann-Whitney U Test.

	Android			iOS		
Scenario	Correct	Incorrect	*p*-value	Correct	Incorrect	*p*-value
Felp	51.3	47.1	0.307	42.1	44.3	0.323
RShare	58.4	66.0	0.028*	56.2	66.4	0.005*
LCGE	73.7	82.7	0.007*	74.3	79.8	0.059

Finding 4.1: *Users who accurately comprehend a permission group, tend to be more conservative in granting it.*

To study the relationship between users' comprehension and their permission decisions, our survey 2 only asked respondents to answer the corresponding permission comprehension questions after they made decisions in the simulated scenarios (§3.3.1). We compared the initial permission grant rates (i.e., the percentage of respondents who allowed the permission request before seeing our provided information) between respondents who correctly and incorrectly answered the comprehension questions. We compared the results of the three scenarios respectively and conducted Mann-Whitney U Test to evaluate the significance of the differences.

As shown in Table 9, users are more likely to deny a permission if they can accurately understand it. The initial grant rates of respondents who correctly answer the comprehension questions are higher in all our evaluated scenarios except for Android users in the scenario of Felp. The differences are evaluated as significant with Mann-Whitney U Test for half of the compared groups. This suggests that when users know exactly what data will be collected, they are more conservative towards granting a permission group. This may protect them from unwanted data exposure or leakages since the data cannot be accessed by the apps in the first place.

As for the Felp (`Contact`) scenario, the Android users who correctly comprehend the scope of the `Contact` permission group accounts for only 6% of all Android respondents. This can cause variability in the results and thus may induce the exceptional results where the grant rate is higher for users who correctly answer the comprehension questions.

5.2 Permission Management

Finding 5: *Users may notice unexpected permissions after reviewing their permission settings yet few of them (two out of 20) regularly review their permission settings.*

In the runtime permission model, users can review and change their permission settings in system settings after the first-time permission decisions. In the interview study, 16 (80%) participants successfully found the permission management interfaces without any guidance from us. We further asked how often they use the permission management and how frequent they review their permission settings. Only two participants mentioned they regularly reviewed their permission settings like every month and would revoke the unnecessary permissions found in the process. Eight participants indicated that they would never review the permission settings. Others just roughly mentioned that they may check the permission settings but not regularly. However, after reviewing their permission settings in our study, five participants quickly noticed permissions unexpectedly granted to some apps. For example, one participant said, *"Why do Whatsapp have access to my location? I don't want anyone to access my location"*.

This finding suggests that few users actively used permission management to revoke unwanted sensitive data access. Previous works explored using personalized privacy nudges to remind users to review settings [26, 48]. They found that many users restricted their permissions after receiving nudges. Recently, Android 10 uses a similar approach to actively remind users if they choose to always allow location access [22]. Future work may look into how to actively engage users in the privacy management without causing habituation [62].

5.3 Developer-Specified Permission Explanations on iOS

Finding 6: *More than half (54.7%) of users did not know that the explanations in the iOS permission dialogs are provided by app developers instead of the system.*

iOS requires app developers to provide explanations for all requested permissions, which will be shown in the permission request dialogs prompted by the system. In the interview, participants were shown the screen of `Camera` permission request for Prisma app on iOS (Figure 7).

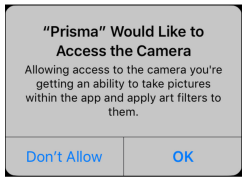


Figure 7: `Camera` request on iOS Prisma app.

The users generally found the explanations helpful in understanding the reasons for the request. However, five participants mistakenly believed that the explanations were provided by the system. We asked the same question in survey 1 to quantify the misunderstanding among iOS users. More than half (54.7%) of iOS respondents believed that the explanations were provided by the system but not by the app developers (47.5%), or chose "I don't know" (7.2%). This indicates that many iOS users confuse app-specified explanations with system-provided information in the permission dialogs.

The interviewees' responses show that several reasons cause their wrong perceptions. First, some users did not believe that app developers have the incentive to help them make permission decisions. Second, the appearance of explanations are consistent across all apps on iOS (e.g. Figure 1(b) and 7). They thought app developers cannot achieve this. Funda-

mentally, this misunderstanding exists because iOS does not explicitly warn the users that the explanations are provided by app developers. Even worse the explanations are displayed in system-provided dialogs without clarification.

This misunderstanding can cause severe problems since users may easily believe the explanations are from trusted and verified sources. Previous research has shown that the developer-specified explanations may contain only partial or inaccurate information on what data will be accessed [50]. To avoid confusion, the iOS system may include the sources of explanations when displaying them in dialogs.

Answer to RQ2: With the limited information provided by system permission request dialogs, users commonly misperceived the scope of permission groups, and had more misperceptions for permission groups controlling a larger number of permissions.

6 Decision Factors

To make permission decisions, users may have concerns and consider more factors other than current information in the permission dialogs. To understand users' concerns and identify the factors that can affect their permission decisions (i.e., *decision factors*), we interviewed 20 mobile users. We further conducted a quantitative study on 1,200 users to evaluate how different decision factors may change users' decisions. The setup of both studies is detailed in Section 3.2 and 3.3.

6.1 Identifying Decision Factors

We define *decision factors* as factors that users may take into account when making permission decisions. In our interview study, we asked the users if they have any concerns when they make permission decisions and what information would help mitigate the concerns. Based on the free-form answers, we concluded five factors in the coding process. The codebooks are in Table 10 and Table 11. To make our study more comprehensive, we also included another factor, the *grant rate* of other users, which was studied by previous work [25,47]. The messages for each decision factor in survey 2 are presented in Table 2 and discussed in §3.3.1.

Finding 7: *Besides **grant rate** studied in previous work [25, 47], users also take other five factors, including **background access, data transmission, brand reputation, rating**, and **review**, into account for making permission decisions.*

Both *internal* and *external factors* can affect permission decisions: Internal factors (i.e. background access and data transmission) illustrate when and how an app will access, transmit or store sensitive data, which can be gathered by systems through monitoring apps' behavior. External factors include brand reputation, rating, review and grant rate. These factors illustrate users' opinions on an app or its producer company. Even though users may have different privacy needs and preferences, previous users' opinions can still provide some insights on privacy usages for the new users. We discuss each factor and the interview study result as follows.

Background access. This factor concerns whether an app will access private data when it is running in the background. It was concerned by *nine* out of 20 participants. In current smartphone OSes, after users grant permission, apps can always access the corresponding data. Therefore, some users were concerned whether an app would abuse the granted permissions to collect private data secretly. Three participants in our study said that they are afraid of apps tracking their locations all the time. For example, P14 said *"Sometimes when I'm talking to others, my phone wakes up and Siri asks me do I need help. Siri, I'm not talking to you. And these home apps, like Alexa, there is some concern."* This also conforms with previous findings that certain background resource accesses are unexpected and uncomfortable for some users [61, 62].

Data transmission. This factor concerns whether apps transmit the collected private data to remote servers. Once the data are transmitted to remote servers, it is unknown how the data will be used. The data can be stored, leaked, or even sold to third parties [14, 24]. *Eight* participants said they are concerned with this factor. For example, P13: *"Like photo-editing app, I expect that they need photo permission because that's what they do. But I am always concerned [that] they collect my photos and do some other things."*

Brand reputation. This factor indicates whether an app's vendor has a good reputation for protecting users' privacy. In our interview study, *eight* participants mentioned that this factor can impact their permission decisions. Some of them are more willing to grant permissions when an app's vendor has a good reputation in protecting privacy (e.g. P15 *"... They (well-known apps) should be more secure. I know who they are, what they do and stuff like this. It makes me easier to give them [requested information]"*). However, there is no gold standard to evaluate companies' reputation and users' evaluation standards can also differ from each other. While many participants mentioned that they are more likely to trust big famous companies, a few (two) participants also showed strong distrust in big tech companies. One participant said that *"I don't trust Facebook at all, they already had all the information and sold it"*. To objectively evaluate a company's reputation, we leveraged the information whether it was known to comply with laws and standards in protecting users' privacy in survey 2 (§3.3)

Ratings and Reviews. These two factors refer to ratings and reviews of an app in app stores. They reflect other users' evaluations of an app's quality. In our study, *15* participants said that they usually look at an app's rating to decide whether to download it or not. Similarly, rating can also be provided to assist users' permission decisions. As P16 puts *"The system can provide something to help users differentiate good apps or bad apps, [that] is helpful, like ratings"*, ratings can help users assess app qualities and may affect permission decisions.

Reviews may contain more detailed descriptions than ratings but only a small proportion of reviews are useful for permission [53, 59]. We analyzed the top 1,000 helpful re-

Table 10: Summary of the interview codebook

Variable	Description	Levels	α [42]
Decision factors	What users concern about and what can mitigate their concerns in granting permissions	See Table 11	0.877
Permission management familiarity	Whether the participants can find the phone setting	Yes/No	1
App store attention.	Relevant information that users read in the app store before users download the apps	Reviews/Ratings/Images/Descriptions	0.942
Permission check frequency	How often the participants check their permissions	Regular/Sometimes/Never	1
Permission model change (Android)	Whether users know about how permission granted for low-version apps in the runtime permission model.	Yes/No	1
Permission explanation provider (iOS)	Provider of the permission explanations in the permission dialog	Systems /App developer/Not sure	1

Table 11: Coding categories for decision factors in the interview study.

Subtheme	Description	Examples
Background access	Participants mention when the app will access the resources in the background	"For microphone, I always concern that some people may listen in my conversations, you know, they have access to that."
Data transmission	Participants mention whether the app will collect/transfer users' data and use it for other things.	"They can get my data, their database may leak my information. The other way is that they can get your data through some network. As long as your data go through the network, there are some risks."
Brand reputation	Participants mention about the app's reputation in security or protecting users' privacy.	"I usually trust the big-companies apps more, [because] I know better about them. They should be more secure."
Rating	Participants mention about ratings of the apps.	"They can provide something to help users differentiate good apps or bad apps, [that] is helpful, like ratings."
Review	Participants mention about reviews of the apps.	"I would like to see some reviews from authorities."

views for a popular free game, "Color Bump 3D", but found only 14 reviews are related to permissions. Thus, it is hard for users to find such information from reviews by themselves. In survey 2, we also presented permission-related reviews to users to see if reviews are helpful to decision making.

Grant rate. This factor refers to what proportion of previous users granted a permission to the same app. Previous work explored the feasibility of crowd-sourcing users' decisions to help users in making permission decisions [25, 47]. The permission settings or privacy expectations from many users were collected and presented to other users when requesting permissions, which has a major impact on users' feelings and their decisions.

6.2 Factors' Impact on Permission Decision

We designed three meaningful scenarios to simulate permission requests from real apps in different contexts. In survey 2, each respondent was provided with only positive or negative messages regarding one decision factor in all three scenarios (§3.3.1) and was asked to rate the helpfulness of the factor. We discuss the major results as follows.

Finding 8: *For the same decision factor, negative messages are more likely to impact users' decisions compared with positive messages.*

The change rates for messages in each scenario are in Table 12. For the negative messages, the change rate is the percentage of participants who changed their decisions from *grant* to *deny* among all participants who initially chose to *grant*. Similarly, for positive messages, the change rate refers to the percentage of respondents who changed from *deny* to *grant*.

We performed Wilcoxon signed rank test to evaluate whether there is a significant difference in user's permission decisions before and after the messages were provided. Table 12 shows the significant change rate of both negative and positive messages in blue background. All negative messages have a significant change rate (p<0.05), but less than half of positive messages have a significant change rate. In addition, most negative messages have a higher change rate than the corresponding positive messages for the same factor, with five exceptions (marked in bold in Table 12).

We used the two-tailed Mann-Whitney U test to measure the differences between the change rates of positive and negative messages. Table 12's p-value columns show the results. Three quarters (27 out of 36) of the change rates are significantly different. These results suggest that negative messages are more likely to impact users' decisions than positive messages. Negative messages may remind users of the potential risks and reconsider their permission decisions.

Interestingly, eight participants changed their decision from grant to deny after being presented with positive messages (*background access* and *data transmission* with LCGE). The reason may be that these participants are very cautious with their location data. Even though the messages are positive, they may be reminded of the potential risks of leaking their locations and thus denied the permission. One participant puts *"Location will be noticed, because that may [have] risks."*

Finding 9: *Users found background access the most helpful while grant rate the least helpful in permission decisions. For the same decision factor, users tended to find the information more helpful if negative messages were shown.*

At the end of survey 2, we asked the respondents to rate the helpfulness of the provided messages. Table 13 shows the results. We computed the average helpfulness score of each factor with positive and negative messages. *Grant rate* has the lowest score: 33 respondents rated this factor as "not helpful at all" (-2). Users would make permission decisions based on their own needs, which may differ from other users'.

Table 12: The change rate for the negative (Neg.) and positive (Pos.) messages for each decision factor. For negative messages, the change rate is the percentage of users who change their decision from 'Allow' to 'Deny', while for positive from 'Deny' to 'Allow'. We mark the change rate in blue if the rate's p-value is significant at $\alpha = 0.05$ in Wilcoxon Signed Rank Test. The column of p-value represents the two-tailed Mann–Whitney U test results of the change rate differences between positive and negative messages.

Scenario	Felp						RShare						LCGE					
Change rate %	Android			iOS			Android			iOS			Android			iOS		
	Neg.	Pos.	p-value	Neg.	Pos.	p-value	Neg.	Pos.	p-value	Neg.	Pos.	p-value	Neg.	Pos.	p-value	Neg.	Pos.	p-value
Background access	46.4	7.7	0.000*	30.3	7.7	0.000*	33.3	15.0	0.004*	16.7	6.7	0.002*	28.6	**41.7**	0.344	46.2	20.0	0.114
Data transmission	24.0	21.4	0.001*	35.0	12.5	0.000*	22.6	**33.3**	0.030*	20.0	12.0	0.000*	10.5	10.0	0.019*	12.8	**46.7**	0.412
Rating	61.5	3.1	0.000*	31.0	20.0	0.007*	50.0	10.0	0.001*	34.4	15.8	0.005*	61.0	41.7	0.349	25.6	**36.4**	0.261
Review	48.0	6.7	0.000*	47.6	5.3	0.000*	39.4	0.0	0.000*	17.2	12.5	0.000*	32.4	23.1	0.075	42.1	0.0	0.008*
Grant rate	37.0	0.0	0.000*	58.3	11.1	0.000*	36.1	5.9	0.002*	42.9	0.0	0.000*	28.9	18.2	0.069	28.6	0.0	0.013*
Brand reputation	52.0	17.9	0.000*	22.6	12.9	0.000*	34.5	33.3	0.079	16.2	**16.7**	0.005*	47.2	28.6	0.117	42.5	15.4	0.040*

Table 13: Helpfulness scores of the decision factors in the negative and positive message framing groups. p-value represents testing result of the helpfulness rating is different between positive and negative group in two-tailed Mann–Whitney U test.

	Negative						Positive						
	+2	+1	±0	-1	-2	avg.	+2	+1	±0	-1	-2	avg.	p-value
Background.	55	28	9	4	4	1.26	41	34	17	6	2	1.06	0.085
Data trans.	25	32	23	12	8	0.54	31	35	19	9	6	**0.76**	0.097
Rating	42	40	11	4	3	1.14	32	37	16	5	10	0.76	0.008*
Review	38	31	13	10	8	0.81	23	35	20	11	11	0.48	0.034*
Grant rate	25	30	14	17	14	0.35	19	26	19	17	19	0.09	0.094
Brand repu.	46	34	14	5	1	1.19	39	35	16	8	2	1.01	0.098

We observe that for the same decision factor respondents found negative messages more helpful than positive messages. For most factors, the average scores of negative messages are much higher than positive messages. This conforms with our previous finding: negative messages are more likely to affect users' permission decisions. For *data transmission*, the negative messages' score is lower than positive messages'. A potential reason is that regarding *data transmission*, the positive messages surprise users more than the negative message. Users may already anticipate their data will be transmitted after collection, in accordance with the negative messages. In contrast, the positive messages break their negative expectations and make them feel comfortable to grant a permission.

We also observe that the helpfulness scores from users who changed their decisions in any of the scenarios (n=345, μ=1.36) are significantly higher than those from users who changed no decision (n=855, μ=0.56) (χ^2 = 368.5, p<0.001). As for demographics, the respondents with experience in computer science or related fields are significantly less likely to change their decisions in the simulated scenarios. (U = 135693.0, p<0.001; two-tailed) No significant difference was observed between the scores from Android and iOS users.

Answer to RQ3: We studied six factors that can affect user's permission decisions: background access, data transmission, brand reputation, rating, review and grant rate, among which, *background access* and *brand reputation* were rated the most helpful by the users. We also found that negative messages related to the factors can have a stronger impact on users' permission decisions.

7 Related work

Install-time permission comprehension. Several works have studied user comprehension of permissions in the install-time permission model [36, 39, 43]. Felt et al. found that most users do not pay attention to the permission notices shown before app installation or do not understand the risks behind the permissions [39]. Kelley et al. found the users can not make informed decisions that based on the technical descriptions for permissions [43]. While these studies shared similar methodologies as our work, they focused on user comprehension in the *install-time model*. Compared with the install-time model, runtime permission dialogs use brief descriptions to describe permission in groups to avoid interrupting users for a long time. This calls for the need to study how users comprehend the permission groups with brief descriptions. We studied this problem with a mixed-methods approach, and found that many users still miscomprehend certain permission groups based on current descriptions (§5).

Felt et al. [36] were among the first to study Android app overprivilege problem where apps request permissions that they do not use. Such problems can be mitigated in the runtime permission model if users can deny the unnecessary permission requests. However, we found that users have misperceptions in certain permission groups based on the information provided by the systems. Therefore, users may not notice such overprivileged apps. This urges the need to improve the design of permission systems and reduce users' misperceptions.

Permission model change. Andriotis et al. [27] studied users' adaptation to the new Android runtime permission model by analyzing the permission settings of 50 users. Their study focused on users' general permission settings as well as users' viewpoint when just adapting to the runtime permission model. Our study focuses on the problem of low-version apps and their prevalence three years after the runtime permission model has been introduced. Surprisingly, we found that low-version apps are still widely installed and one-third users have confusion on their behavior of requesting permissions (§4).

Rationale messages in requesting permissions. Previous works studied the rationale messages provided by app devel-

opers [30, 50, 57]. Bonné et al. [30] found that users grant or deny a permission based on their expectation on whether an app needs the permission. Both Android and iOS adopt the practice to let app developers provide rationale messages to explain how permissions are used [16, 19]. However, only relying on app developers providing rationale messages suffer from several problems [50, 57]. First, app developers may not provide correct and helpful rationale messages. Liu et al. [50] found that a significant portion of incomplete explanations only describe basic permissions but hide their usage of other permissions in the same permission group. Second, Tan et al. [57] found that most messages only focus on the user benefits but not the potential risks. In comparison, we moved one step further to investigate what *systems* can provide to help users understand permissions (§6).

User concerns in granting permissions. Many previous works aim to understand what concerns users have when granting permissions [35, 38, 44, 61]. Inspired by these work, our study aims to explore what additional information (factors) that systems can provide to resolve users' concerns and assist them in making permission decisions.

Felt et al. [38] surveyed and ranked users' concerns on risks related to private data that can be accessed by apps. Their research goal lies in the selection of private data that should be protected by permissions and warned to users. Our study complements their work by focusing on what information can be provided by the systems to improve users' understanding of the permission requests and address their concerns.

Other related works cover certain aspects of the five identified decision factors. (1) Previous works [28, 35] proposed program analysis techniques that can detect sensitive *data transmission* in Android apps. These techniques can be helpful in understanding application sensitive data usage behaviors and derive the information related to decision factors. (2) Previous research on the impact of *background access* shows that users are more likely to be uncomfortable with resources requested in the background and block the requests [51,61,62]. Votipka et al. [61] found that users' comfort level of the background resource access depends on the *when* and *why* the resource was used. As a complement, our quantitative study shows that *background access* is rated as the most helpful one among the six decision factors. (3) Previous works found that app store information of user *rating* and *reviews* have significant impact on both apps' improvement [53] and users' decision on updating apps [59], but none of them have explored whether *ratings* and *reviews* can help users' permission decisions. (4) We found no previous work studied the relationship between *brand reputation* and users' permission decisions. (5) We also included grant rate as one decision factor based on previous studies [25, 47]. Lin et al. used the percentage of users that find a permission surprising to remind users at the install-time warnings [47]. Agarwal et al. [25] used the collected *grant rate* to make permission recommendations for new users However, we found that *grant rate* is rated as the least helpful among the six decision factors, even though this factor will impact many users' permission decisions (§6.2).

Other previous works in HCI communities explored the feasibility to incorporate additional information to raise users' attention to privacy and permissions [44] or help users better understand permissions through examples [41]. These works focused more on *how* to present the information to users;Our paper studied *what* should be presented to the users by comparing different decision factors (§6) .

Context integrity for mobile privacy. Context integrity [54] ties privacy protection to specific contexts. Wijesekera et al. [62] found that users may make different decisions for the same permission when it is used in different contexts and further proposed a machine learning approach that leverages users' past permission decisions to predict future decisions each time when permissions are used [63]. Tsai et al. [60] proposed a context-aware permission manager to help users flexibly control data access, e.g. only allow data access when the app is in the foreground. Different contexts can indeed affect users' permission decisions. Our study confirms that background access (or visibility in [62, 63]) can affect permission decisions. However, we focus on more general questions: (1) whether users can comprehend the permission groups and their related security risks, and (2) what additional information can be provided to enhance user comprehension. Our work is significant concerning context integrity. First, it is essential to ensure that users make permission decisions when they can understand the permissions. This is the case especially when considering context integrity where users' future permission decisions can be made based on their previous ones [63]. Second, we identified decision factors that can affect permission decisions other than the contexts defined by Wijesekera et al. [62]. These factors may also be used as features to improve privacy decision prediction model to make permission decisions aligning with user preferences [63].

Permission Fatigue. Previous works found that repetitive warnings can lead to notice fatigue and habituation in software agreement notices [40], Android install-time notices [37, 39] and browser security warnings [56]. Our findings provide hints for addressing the fatigue problem (§6). In order to improve user attentions, systems may allow users to customize decision factors based on their preferences. In addition, systems can highlight negative messages to draw users' attention. Bravo-Lillo et al. found that forcing users interact with essential information or adding attractors can effectively increase user attention and address habituation [32, 33]. Similarly, our findings in §5.3 suggest that it is necessary to highlight several important information (e.g. the provider of permission explanations) to draw users' attention.

8 Discussion and Implication

8.1 Explaining permission model changes

In §4, our study disclosed the prevalence of low-version Android apps as well as users' common misunderstanding

on these low-version apps. Low-version apps get all permissions at install-time even on new systems, while most users mistakenly expect they request permissions at runtime. This misunderstanding may be potentially taken advantage to bypass users' consent at runtime and cause privacy leakage.

At the same time of our study, two efforts have been made to mitigate this issue. First, Google Play begins to disallow uploading low-version apps [11]. Second, Android 10 asks user to decide to revoke dangerous permissions or not when the low-version app launches for the first time [22]. However, these efforts may not resolve the issue in total. First, many low-version apps uploaded before may still exist on Google play or third-party app stores. Second, the adoption of Android 10 may take a long time. Users may still be impacted by low-version if their OSes are not updated to Android 10.

We hope our study can raise people's awareness of this issue and inspire future works. First, third-party app stores may also consider forbidding low-version apps as Google Play did. Second, system designers may consider giving explicit warnings of low-version apps to raise users' attention. Third and more fundamental, system designers may consider to examine the comparability mechanism carefully to avoid similar issues which may confuse users.

8.2 Addressing common misunderstandings for permission groups

Our findings in §5 indicate that users commonly misunderstood the scope of permission groups. We suggest two potential approaches to reduce misunderstandings: OSes can (a) provide more explanations in the permission dialog, or (b) reorganize the permission groups and make them more intuitive (e.g., breaking down the permission groups into smaller ones). However, long explanations or excessive permission requests during runtime increase users' recognition burdens, making them habituate and ignore the explanations [29,37,39]. Future work could therefore explore how best to provide more explanations while balancing their complexity.

Our findings in §5.3 show that many iOS users commonly confuse app explanations with system-provided information. In addition, previous work found that many app-specified explanations only focus on user benefits and provide inaccurate information on what data will be accessed [50,57]. It is worthwhile to study how to assist app developers in providing better explanations and how to audit such explanations against app behaviors to avoid misleading users.

8.3 Addressing concerns with decision factors

In §6, we studied six factors that can affect users' permission decisions. We observed that negative messages of the factors are considered more helpful and more likely to affect user decisions than the positive ones. Following our findings, future work can focus on how to extract information concerning the factors. Here, we discuss some potential approaches. Internal factors (i.e., background access and data transmission) can be collected by OSes. For example, background access can be tracked by logging related system APIs. Data transmission can be monitored by combining static and dynamic data flow analysis [28]. The information can be collected during testing or in real use. Recent updates in Android [2] and iOS [4] allow users to choose whether to grant background access to locations, which notifies users about the potential background access.

External factors (i.e. rating, review, grant rate and brand reputation) can be collected via crowdsourcing or through trusted organizations with efforts. Reviews and ratings are available in app stores. OSes can collect them via information retrieval techniques and present it to users. Grant rate can be collected by the phone vendors from their users. Existing security and privacy standards like ISO/IEC27001 or GDRP [3, 5] may be used to reflect the company's reputation in protecting users' privacy. The general challenge concerning external factors is that information of these factors may be manipulated by fraudulent third parties. More efforts are needed to ensure that the obtained information is trusty.

Our work mainly focuses on the simple variations of the messages for decision factors, namely positive and negative messages. Future work may study more fine-grained metrics on three decision factors (rating/review/grant rate), e.g. how specific values can impact users' decisions. Our study also shows that users' decisions may change differently for different decision factors (Finding 9), which aligns with previous study that users have different privacy needs [48, 49]. This implies the necessity for a framework which allows users to personalize and configure what information to be provided. Future work may further study users' capability and willingness to configure these decision factors.

9 Conclusion

Current mobile systems play a neutral role in protecting users' private information—they just provide simple descriptions and allow apps to explain their permission request intentions. This can easily lead to unintended privacy leakage because of users' poor understandings of the permissions. In this paper, we investigated the problem through analysis of real users' permission settings and large-scale user studies. We find that users have several common misunderstandings on certain permission groups and many Android users are not aware of permission model changes. This motivates system designers to enhance systems by providing clearer permission-related information. We further studied what extra information can be provided by the systems to help users make more informed decisions. Our results suggest that information about *background access* and *brand reputation* were rated the most helpful and the negative messages related to the factors can have a stronger impact on users' decisions. Such results can guide system designers to select relevant information that can raise users' attention when making permission decisions.

Acknowledgments

We greatly appreciate the anonymous reviewers for their insightful comments and feedback. We thank Shelby Thomas, C. Ailie Fraser, Vector Guo Li and a host of others in the Opera group, the Systems and Networking group at UC San Diego and Whova Inc for useful discussions and paper proofreading. This work is supported in part by NSF grants (CNS-1814388, CNS-1526966) and the Qualcomm Chair Endowment. Lili Wei was supported by the Postdoctoral Fellowship Scheme by the Hong Kong Research Grant Council.

References

[1] Android fragmentation keeps getting worse. https://tinyurl.com/yd78kncd.

[2] Android location updates. https://developer.android.com/preview/privacy/location.

[3] General data protection regulation gdpr. https://gdpr-info.eu/.

[4] iOS 13 location updates. https://gimbal.com/ios-13-location-permissions/.

[5] Iso/iec 27001 information security management standard. https://www.iso.org/isoiec-27001-information-security.html.

[6] List of android app stores. https://en.wikipedia.org/wiki/List_of_Android_app_stores.

[7] Permission checker dataset. https://ucsdopera.github.io/PermissionStudyUsenix21/dataset/.

[8] Permission checker privacy policy. https://permissionchecker.github.io/privacy.html.

[9] Permission checker website. https://permissionchecker.github.io.

[10] Supplementary materials. https://ucsdopera.github.io/PermissionStudyUsenix21/supplementary.pdf.

[11] Target api level requirements for the play console. https://tinyurl.com/y6uu6saz.

[12] ios 6 to seek permission before apps can access personal data. https://tinyurl.com/yxhz7pz4, 2012.

[13] Meitu has major privacy red flags. https://tinyurl.com/zd5z5l7, 2017.

[14] 7 in 10 smartphone apps share your data with third-party services. https://tinyurl.com/ybe46d3c, 2018.

[15] Android permission overview. https://developer.android.com/guide/topics/permissions/overview, 2018.

[16] Android request app permissions. https://developer.android.com/training/permissions/requesting, 2018.

[17] Facebook has been collecting call history and sms data from android. https://tinyurl.com/yb5vkngd, 2018.

[18] Gobuff record and send screen recordings in the background. https://tinyurl.com/yyzrmbxq, 2018.

[19] ios accessing protected resources. https://tinyurl.com/ufhfe7c, 2018.

[20] Pokemon go abuse storage read permission to combat rooting. https://tinyurl.com/y3qkobc9, 2018.

[21] Android official documentation: Requesting permissions at runtime. https://developer.android.com/training/permissions/requesting.html, 2019.

[22] Android q privacy update. https://developer.android.com/about/versions/10/privacy/changes, 2019.

[23] Smartphone users will top 3 billion in 2018, hit 3.8 billion by 2021. https://tinyurl.com/y4mxrqaw, 2019.

[24] Ring for Android reportedly shares your data with third parties. https://mashable.com/article/ring-third-party-data/, 2020.

[25] Yuvraj Agarwal and Malcolm Hall. Protectmyprivacy: detecting and mitigating privacy leaks on ios devices using crowdsourcing. In *Proc. MobiSys*, pages 97–110. ACM, 2013.

[26] Hazim Almuhimedi, Florian Schaub, Norman Sadeh, Idris Adjerid, Alessandro Acquisti, Joshua Gluck, Lorrie Faith Cranor, and Yuvraj . Your location has been shared 5,398 times!: A field study on mobile app privacy nudging. In *Proc. CHI*, pages 787–796, 2015.

[27] Panagiotis Andriotis, Martina Angela Sasse, and Gianluca Stringhini. Permissions snapshots: Assessing users' adaptation to the android runtime permission model. In *2016 IEEE WIFS*, pages 1–6. IEEE, 2016.

[28] 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. *Acm Sigplan Notices*, 49(6):259–269, 2014.

[29] Rainer Böhme and Jens Grossklags. The security cost of cheap user interaction. In *Proc. New Security Paradigms Workshop*, 2011.

[30] Bram Bonné, Sai Teja Peddinti, Igor Bilogrevic, and Nina Taft. Exploring decision making with android's runtime permission dialogs using in-context surveys. In *Proc. SOUPS*, 2017.

[31] Virginia Braun and Victoria Clarke. Using thematic analysis in psychology. *Qualitative research in psychology*, 2006.

[32] Cristian Bravo-Lillo, Lorrie Cranor, Saranga Komanduri, Stuart Schechter, and Manya Sleeper. Harder to ignore? revisiting pop-up fatigue and approaches to prevent it. In *Proc. SOUPS*.

[33] Cristian Bravo-Lillo, Saranga Komanduri, Lorrie Faith Cranor, Robert W Reeder, Manya Sleeper, Julie Downs, and Stuart Schechter. Your attention please: designing security-decision uis to make genuine risks harder to ignore. In *Proc. SOUPS*, pages 1–12, 2013.

[34] Permission Checker. Permission checker on google play. https://play.google.com/store/apps/details?id=com.sbysoft.perchecker, 2018.

[35] 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*, 2014.

[36] Adrienne Porter Felt, Erika Chin, Steve Hanna, Dawn Song, and David Wagner. Android permissions demystified. In *Proc. CCS*. ACM, 2011.

[37] Adrienne Porter Felt, Serge Egelman, Matthew Finifter, Devdatta Akhawe, David Wagner, et al. How to ask for permission. In *HotSec*, 2012.

[38] Adrienne Porter Felt, Serge Egelman, and David Wagner. I've got 99 problems, but vibration ain't one: a survey of smartphone users' concerns. In *Proc. of 2nd ACM workshop on Security and privacy in smartphones and mobile devices*, 2012.

[39] Adrienne Porter Felt, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner. Android permissions: User attention, comprehension, and behavior. In *Proc. SOUPS*, page 3. ACM, 2012.

[40] Nathaniel S Good, Jens Grossklags, Deirdre K Mulligan, and Joseph A Konstan. Noticing notice: a large-scale experiment on the timing of software license agreements. In *Proc. CHI*, pages 607–616, 2007.

[41] Marian Harbach, Markus Hettig, Susanne Weber, and Matthew Smith. Using personal examples to improve risk communication for security & privacy decisions. In *Proc. CHI*, pages 2647–2656. ACM, 2014.

[42] Andrew F Hayes and Klaus Krippendorff. Answering the call for a standard reliability measure for coding data. *Communication methods and measures*, 1(1):77–89, 2007.

[43] 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 *Intl. conf. on financial cryptography and data security*. Springer, 2012.

[44] Patrick Gage Kelley, Lorrie Faith Cranor, and Norman Sadeh. Privacy as part of the app decision-making process. In *Proc. CHI*. ACM, 2013.

[45] Frauke Kreuter, Stanley Presser, and Roger Tourangeau. Social desirability bias in cati, ivr, and web surveysthe effects of mode and question sensitivity. *Public opinion quarterly*, 2008.

[46] Jon A Krosnick. Survey research. *Annual review of psychology*.

[47] Jialiu Lin, Shahriyar Amini, Jason I Hong, Norman Sadeh, Janne Lindqvist, and Joy Zhang. Expectation and purpose: understanding users' mental models of mobile app privacy through crowdsourcing. In *Proc. Ubicomp*. ACM, 2012.

[48] Bin Liu, Mads Schaarup Andersen, Florian Schaub, Hazim Almuhimedi, Shikun Aerin Zhang, Norman Sadeh, Yuvraj Agarwal, and Alessandro Acquisti. Follow my recommendations: A personalized privacy assistant for mobile app permissions. In *Proc. SOUPS*, 2016.

[49] Bin Liu, Deguang Kong, Lei Cen, Neil Zhenqiang Gong, Hongxia Jin, and Hui Xiong. Personalized mobile app recommendation: Reconciling app functionality and user privacy preference. In *Proc. WSDM*, 2015.

[50] Xueqing Liu, Yue Leng, Wei Yang, Wenyu Wang, Chengxiang Zhai, and Tao Xie. A large-scale empirical study on android runtime-permission rationale messages. In *IEEE VL/HCC*, pages 137–146. IEEE, 2018.

[51] Kristopher Micinski, Daniel Votipka, Rock Stevens, Nikolaos Kofinas, Michelle L Mazurek, and Jeffrey S Foster. User interactions and permission use on android. In *Proc. CHI*, pages 362–373. ACM, 2017.

[52] Muhammad Baqer Mollah, Md Abul Kalam Azad, and Athanasios Vasilakos. Security and privacy challenges in mobile cloud computing: Survey and way ahead. *Journal of Network and Computer Applications*, 84:38–54, 2017.

[53] Duc Cuong Nguyen, Erik Derr, Michael Backes, and Sven Bugiel. Short text, large effect: Measuring the impact of user reviews on android app security and privacy. In *IEEE S&P*. IEEE, 2019.

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

[55] Elissa M Redmiles, Ziyun Zhu, Sean Kross, Dhruv Kuchhal, Tudor Dumitras, and Michelle L Mazurek. Asking for a friend: Evaluating response biases in security user studies. In *Proc. CCS*, pages 1238–1255. ACM, 2018.

[56] Joshua Sunshine, Serge Egelman, Hazim Almuhimedi, Neha Atri, and Lorrie Faith Cranor. Crying wolf: An empirical study of ssl warning effectiveness. In *USENIX security symposium*, pages 399–416, 2009.

[57] Joshua Tan, Khanh Nguyen, Michael Theodorides, Heidi Negrón-Arroyo, Christopher Thompson, Serge Egelman, and David Wagner. The effect of developer-specified explanations for permission requests on smartphone user behavior. In *Proc. CHI*, pages 91–100. ACM, 2014.

[58] Christopher Thompson, Maritza Johnson, Serge Egelman, David Wagner, and Jennifer King. When it's better to ask forgiveness than get permission: attribution mechanisms for smartphone resources. In *Proc. SOUPS*, pages 1–14, 2013.

[59] Yuan Tian, Bin Liu, Weisi Dai, Blase Ur, Patrick Tague, and Lorrie Faith Cranor. Supporting privacy-conscious app update decisions with user reviews. In *Proc. ACM CCS Workshop on Security and Privacy in Smartphones and Mobile Devices*, pages 51–61, 2015.

[60] Lynn Tsai, Primal Wijesekera, Joel Reardon, Irwin Reyes, Serge Egelman, David Wagner, Nathan Good, and Jung-Wei Chen. Turtle guard: Helping android users apply contextual privacy preferences. In *Proc. SOUPS*, 2017.

[61] Daniel Votipka, Seth M Rabin, Kristopher Micinski, Thomas Gilray, Michelle L Mazurek, and Jeffrey S Foster. User comfort with android background resource accesses in different contexts. In *Proc. SOUPS*, pages 235–250, 2018.

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

[63] Primal Wijesekera, Arjun Baokar, Lynn Tsai, Joel Reardon, Serge Egelman, David Wagner, and Konstantin Beznosov. The feasibility of dynamically granted permissions: Aligning mobile privacy with user preferences. In *IEEE S&P*, pages 1077–1093. IEEE, 2017.