# I don't know what I've all granted. Does it really matter? – Understanding Users' Awareness of Different Permission Types on Android

Verena Winterhalter
*LMU Munich, Germany*
*verena.winterhalter@ifi.lmu.de*

Sarah Prange
*LMU Munich, Germany*
*sarah.prange@ifi.lmu.de*

Anouk Moreno
*LMU Munich, Germany*
*anouk.moreno@campus.lmu.de*

Harel Israel Berger
*Ariel University, Israel*
*harelb@ariel.ac.il*

Florian Alt
*LMU Munich, Germany*
*University of the Bundeswehr Munich, Germany*
*florian.alt@ifi.lmu.de*

## Abstract

In this paper, we present the results of an in-the-wild study ($N = 77$) using an Android study app that captures real permission configurations, in-situ experience sampling of participants' perceptions of which permissions apps hold, and risk awareness for permission combinations during everyday use. Such understanding matters because Android attacks can exploit individual permissions and, in particular, risky combinations. At the same time, it is unclear how often users grant such permissions (and combinations) and whether they are aware of this threat. We found that users frequently misjudge app permission states, regardless of permission type. This also holds for permissions that form part of risky combinations. 56.76% of responses mismatched the actual permission state, including 60.70% for Runtime permissions, 51.19% for Install-time permissions, and 47.67% for permissions which are part of risky combinations reported in prior security literature. Because awareness of granted permissions is a prerequisite for informed action, these gaps hinder users' ability to mitigate privacy and security risks. We discuss the implications for permission user interfaces and platform mechanisms to better surface risks arising from permission combinations.

## 1 Introduction

Permission decisions on smartphones are often framed as isolated, one-off prompts. Yet users' privacy and security depend on the *current permission state* of their apps – what access is actually granted right now – shaped by prompts, defaults, revocations, and changes over time. In everyday use, this state is easy to lose track of: permissions are granted in moments of need, forgotten afterward, and rarely revisited. Some permissions are requested once at install time, others can be toggled or revoked later, and app updates or changes in user behavior can make previously "reasonable" access newly problematic. As a result, users may hold beliefs about what an app can access that no longer match reality, and they may not notice when access accumulates across apps or over time.

Prior research has repeatedly shown that users struggle to understand, predict, and manage app permissions [17]. Much of this evidence comes from self-reports, one-shot surveys, or lab studies that cannot fully capture permission behavior as it unfolds in everyday life – contextual, longitudinal, and shaped by the privacy paradox [20]. At the same time, the Android permission system has evolved to offer more granular control (e.g., Runtime granting and revocation) [3, 5]. Despite these improvements, users often remain unaware of which permissions are currently active, how they ended up in that state, or how to change it [7, 14, 19, 29]. System-level dynamics further complicate this: app updates can effectively expand access (e.g., via permission groups) and reduce transparency [11], while permission explanations can be vague or misleading [12]. Together, these factors create a permission ecosystem where users are formally empowered, yet practically disempowered in understanding and managing access.

The practical risk of smartphone permissions is determined by what apps can access *in practice* – that is, the permissions they currently hold on a user's device. Importantly, harmful outcomes can stem not only from individual permissions but also from the *accumulation* of multiple permissions that together enable more powerful actions (e.g., surveillance or manipulation) than any single permission alone. Android malware exploits permissions [26], e.g. using accessibility[1] or NFC permissions[2] for malicious activities.


*USENIX Symposium on Usable Privacy and Security (SOUPS)* 2026.
August 23–26, 2026, Hannover, Germany.

[1] https://www.malwarebytes.com/blog/news/2025/12/new-android-malware-lets-criminals-control-your-phone-and-drain-your-bank-account. Last accessed: 2026-06-10.

[2] https://cybersecuritynews.com/new-ghost-tapped-attack-uses-your-android-device/. Last accessed: 2026-06-10.

Prior work used permission signals to detect malware and characterize potentially malicious behavior [9,27]. From a human perspective, we still know little about whether users can accurately recognize which permissions apps currently have, how often their beliefs about current access match reality, and what factors predict these mismatches. Answering these questions requires observing permissions as they exist and change in the wild – what users grant, revoke, or ignore over time – alongside users' perceptions at the moments those configurations matter. Traditional methods struggle to combine longitudinal behavioral measurement and situated perception.

To address this gap, we conducted an in-the-wild study (N=77) using *PermWatch* [33], an Android research app that (1) logs participants' real permission configurations over time and (2) samples participants in situ about which permissions they believe selected apps currently have. We compare reported vs. actual permission states across runtime and install-time permissions, including a subset of permissions highlighted in prior work on risky combinations, and analyze factors associated with correct vs. incorrect judgments. Our goal is not to test whether users can enumerate "risky combinations" per se, but to measure whether they can correctly identify the permissions that constitute current access and enable risk.

We found that participants frequently misjudged which permissions apps currently held (total mismatched answers: 56.76%) and often expressed uncertainty (total "Don't know answers": 31.55%), indicating that many users lack a reliable understanding of ongoing access on their own devices. Misconceptions were systematic: Runtime permissions yielded many False Positives (believing access was granted when it was not, 23.95%), whereas Install-time permissions yielded many False Negatives (believing access was denied despite being granted by default, 21.63%). Correctness was associated with whether a permission was granted and with app usage recency, suggesting "unseen" permissions and rarely used apps as key targets for improving permission transparency.

These findings have two key implications. First, permission interfaces that primarily support one-time, single-permission decisions leave users with persistent blind spots about ongoing access – especially for less visible permission types and less frequently used apps. Second, supporting user-aligned decision-making may require interventions that make current permission states more legible and actionable over time (e.g., beyond one-time prompts). Systems that include better cues for reviewing and revisiting permissions as apps and usage contexts change keep the user in the decision loop in a feasible way, giving them autonomy over their permission settings.

**Contribution Statement.** We contribute: (1) Empirical data on installed apps and permission states on Android phones, revealing essential mismatches between granted permissions and user perception/response in our dataset. (2) An analysis of factors that influence likelihood of user estimation being correct, and make suggestions how these findings can inform the design of future interfaces and privacy interventions.

## 2 Background & Related Work

In this section, we explore known works on Android permissions and associated security risks. We also explore prior user studies on permissions across different mobile platforms.

### 2.1 Android Permissions

Android permissions regulate applications' access to system resources, sensors, and user data, and are associated with protection levels (e.g., *normal* and *dangerous*) in the Android security model. From a security perspective, risks may arise from individual permissions and from the accumulation of multiple permissions that jointly enable more powerful actions. For our purposes, the key point is that apps' *current permission states* are not necessarily salient to users and can differ from users' beliefs, motivating our focus on permission-state awareness in everyday use. Android distinguishes between permissions that are granted by default and those that require explicit user approval and can later be changed in settings [3,5]. Hence, an app's effective access can evolve over time through user actions, system behavior, or app updates.

Several studies have demonstrated the effectiveness of permission combinations for Android malware detection. *PermPair* [9] introduced a detection approach based on pairs of permissions and identified ten combinations that serve as significant indicators for detecting malicious applications, such as `ACCESS_NETWORK_STATE` with `READ_PHONE_STATE`, or `READ_PHONE_STATE` with `WRITE_EXTERNAL_STORAGE`. Similarly, Wang et al. [31] quantified the risk associated with permission sets by leveraging Android's official protection levels [2], showing that certain co-occurring permissions are strong indicators of malicious behavior.

More recent work has focused on ranking the contribution of individual permissions and small permission subsets within machine-learning-based detectors. IPAnalyzer [28] identified a set of ten highly influential permissions, such as `BIND_GET_INSTALL_REFERRER_SERVICE`, that achieved strong detection performance when combined. While IPAnalyzer also examined intents and their interaction with permissions, such features are outside the scope of this work, as intents cannot be revoked by users at runtime.

Beyond detection, prior research highlighted permission combinations that directly enable concrete attack vectors. Guerra-Manzanares et al. [21] showed that permissions such as `SEND_SMS`, `READ_PHONE_STATE`, `INTERNET`, and `MOUNT_UNMOUNT_FILESYSTEMS` remained consistently significant within the Android ecosystem over time. Fratantonio et al. [18] demonstrated that the combination of `SYSTEM_ALERT_WINDOW` and `BIND_ACCESSIBILITY_SERVICE` enables stealthy credential theft, PIN harvesting, and the installation of a so-called *God-mode* application with near-complete device control. Rahman et al. [25] categorized dangerous permission groups,

including permissions that allowed access to persistent identifiers, storage access, and location-related permissions.

Permissions have also played a central role in adversarial and evasion-based attacks. Berger et al. [10] showed that modifying permission declarations can mislead multiple malware detection systems and VirusTotal scanners[3]. Tuncay et al. [30] revealed security threats introduced by custom Android permissions, which are often insufficiently analyzed by automated tools. ShadowDroid [35] leveraged permissions alongside other static features to generate adversarial malware samples that evade state-of-the-art detectors.

Overall, permission combinations are powerful indicators of malicious behavior and enablers of real-world attacks. Nevertheless, most prior studies focus on detection accuracy or adversarial robustness, while comparatively little attention has been paid to whether users can accurately recognize what permissions apps currently hold on their own devices and how such beliefs drift from reality as permission states evolve over time. Our work builds on the technical background above, focusing on this user-centered gap between permission *ground truth* and beliefs about ongoing access in real-world use.

## 2.2 User Studies – Permission Risks

A substantial body of user-centered research shows that users frequently misunderstand how mobile applications access sensitive resources and how such access translates into privacy risk. Early work by Wijesekera et al. [32] demonstrated that Android users are often surprised by apps' access to private resources and, retrospectively, would have preferred to restrict such access. Recent studies reinforce this finding: Prange et al. [24] provide a detailed analysis of users' motivations for revoking Android permissions, highlighting that revocation decisions are often reactive and shaped by unexpected app behavior rather than by an accurate understanding of threats.

Beyond simple awareness, prior work shows that how permission risks are communicated strongly influences user decisions. Zhang et al. [36], studying iOS users, find that visual, grid-based representations of data flows significantly improve users' ability to identify privacy leaks compared to traditional policy-based disclosures. While conducted on iOS, these results suggest that users' difficulties might stem less from platform-specific mechanisms and more from general limitations in mental models of permission-mediated data access. Similarly, Distler et al. [13] emphasize the diversity of risk representations, from simulated to reported consequences, as well as methodological choices such as task-driven exploration or deception to surface privacy-relevant behavior.

Large-scale surveys further indicate that many users remain disengaged from permission management altogether. Frik et al. [19] report that a significant fraction of users are unaware of available privacy controls or refrain from modifying them due to low perceived expertise or self-efficacy. Complementary analyses reveal heterogeneity in permission behavior: Alsoubai et al. [8] identify distinct user profiles based on installed apps and granted permissions, building on earlier conceptualizations of "privacy personas" [15]. Together, these suggest that users' permission decisions are shaped by stable behavioral tendencies as much as by situational factors.

Recent studies deepen this perspective by examining psychological and contextual factors that influence permission decisions. Research on permission rationales shows that linguistic framing, such as politeness or the emphasis on negative consequences of denial, can bias users toward granting access, independent of objective risk [16]. Affective studies demonstrate that perceived intrusiveness varies with context: for instance, microphone access is judged more invasive at home than in public settings [34]. To counter abstract or opaque warnings, visual consequence systems illustrate potential data misuse through animations, which are assumed to be effective for younger users [23]. Finally, emerging work highlights persistent gaps in users' mental models for sensors they are not familiar with, compared to more well-known permissions [6].

In summary, users struggle to reason about permission risks, their decisions are sensitive to framing and context, and awareness alone does not translate into informed control. However, the literature largely focuses on individual permissions or isolated access decisions, and only indirectly reflects how users reason about the combined implications of multiple granted permissions. Our work builds on these insights, examining how users' qualitative perceptions of permission combinations align, or diverge, from the actual risks posed by multi-permission attack vectors on Android.

# 3 Research Approach

In this work, we investigate how users' *actual* permission configurations on their smartphones relate to their *self-reported* understanding and awareness of those permissions, with a focus on different permission types (Runtime vs. Install-time permissions, and risky permission sets, cf. Section 3.1). Specifically, we address the following research questions:

- **RQ1: Self-Reported Awareness:** How aware are users of permissions and privacy implications, in particular for dangerous permission combinations?
- **RQ2: Misconceptions:** How do actual permission states deviate from self-assessment? What are misconceptions?

To answer these questions, we use *PermWatch*, a research application that enables in-the-wild, longitudinal collection of (i) actual permission states configured on participants' phones, and (ii) experience-sampling responses about participants' perceived permission status and privacy implications [33]. We then compare logged permission configurations against self-reports to quantify awareness and misconceptions.

[3] https://www.virustotal.com. Last accessed: 2026-02-18.

## 3.1 Permission Types

The different permission types we looked at were *Runtime*, *Install-time*, and *Permissions in Risky Permission sets*:

**Runtime permissions** These are permissions where the user gets prompted during runtime [3], e.g., while they use the app, to make a decision whether they want to grant it or not. Depending on the Android version, this might include options such as "Only allow this time". Runtime permissions can either be currently granted or denied. As Runtime permissions are only requested from the user when certain functionalities associated with the permission are accessed, a "denied" permission state could either indicate that the user actively declined this permission when prompted to grant it, that the "*auto revoke*" mechanism [1] of Android revoked the permission once the app was not used for a longer time period, or that the user has not yet encountered the request.

**Install-time permissions** These permissions are granted when an app is installed [3], i.e. without an explicit prompt presented to the user. Usually, the page in the app store provides information about the required Install-time permissions. If an app declares permissions in its manifest, they are granted as long as the app is installed.

**Risky Permission Sets / PermPair** The ten risky permission combinations identified by Arora et al. [9] are made up of six individual permissions, including two Runtime and four Install-time permissions. We call the six permissions that are part of the risky combinations "PermPair Permissions". Table 1 contains the full set of PermPair combinations.

## 3.2 Data Collection Method

During the study, we collected two types of data:

**Automated Permission Logging** During the initial setup, and once per day when the daily questionnaire started, we scanned participants' phones for all installed applications, respective permissions, and states.

| PermPair Permission 1 | PermPair Permission 2 |
|---|---|
| INTERNET | ACCESS_NETWORK_STATE |
| INTERNET | WAKE_LOCK |
| INTERNET | READ_PHONE_STATE* |
| INTERNET | ACCESS_WIFI_STATE |
| WRITE_EXTERNAL_STORAGE* | INTERNET |
| WRITE_EXTERNAL_STORAGE* | ACCESS_NETWORK_STATE |
| WRITE_EXTERNAL_STORAGE* | READ_PHONE_STATE* |
| ACCESS_NETWORK_STATE | WAKE_LOCK |
| ACCESS_NETWORK_STATE | READ_PHONE_STATE* |
| ACCESS_NETWORK_STATE | ACCESS_NETWORK_STATE |

Table 1: Overview of the PermPair combinations. Runtime permissions are marked with *.

**User Feedback** We gathered users' self-reports through several questionnaires:

- *Initial Questionnaire* covering basic demographics.
- *Daily Questionnaires ("Dailies")* covering permissions for selected apps (see below).
- *Final Questionnaire* on general knowledge about permissions and privacy implications.

For the *daily questionnaires*, we selected apps from participants' phones as follows: We generated a pool of 30 apps by selecting the first and last ten apps when sorting by the "last used" timestamp. The remaining ten apps were selected randomly from the remaining apps (e.g., not the ten most recently/least recently used apps). We made sure all of these are user-facing apps, i.e., have an app launcher that users can interact with and are not declared as system apps by Android. We refer to these categories as *"most recently"*, *"least recently"*, and *"occasionally"* used.

For daily questionnaires, three apps were selected (one per category), resulting in 21 apps covered by all daily questionnaires[4]. For each app, we asked for ten permissions, whether participants believed these were granted to that app: *"Which of these permissions does the app currently have access to?"*. Permissions were selected as follows: if a PermPair permission was present in an app's manifest, it was included in the questionnaire. Next, we chose up to two Runtime permissions and up to two Install-time permissions that the app had declared. If we did not fill 10 permissions by then, we randomly added others declared in the app's manifest[5].

## 3.3 Procedure

The detailed procedure of our study was as follows (see Figure 1 for an overview, and Appendix A for the installation guide and full list of questions):

1. *App Installation:* Participants downloaded and installed our study app. Upon setup, it asked for informed consent and permission to access app data and send notifications.
2. *Initial Questionnaire:* Participants filled an initial questionnaire on demographics.
3. *Daily Questionnaire:* Participants filled a total of seven questionnaires, with the first questionnaire following directly after the installation and initial questionnaire. After that, participants received notifications once per day to complete a new questionnaire until a total of seven were completed. If they skipped a day without filling out the questionnaire, the study duration would increase, since the questionnaire was presented the next day.

[4]The additional apps were selected as fallback options if users delete apps during the study. When selecting the apps for the day, the study app checks if the selected apps are still installed on the phone, and otherwise discards and replaces the uninstalled app with another one from the same category.

[5]In a few cases where apps declared fewer than 10 permissions in their manifest, we filled the questionnaire with random permissions. We later excluded those apps from the analysis.

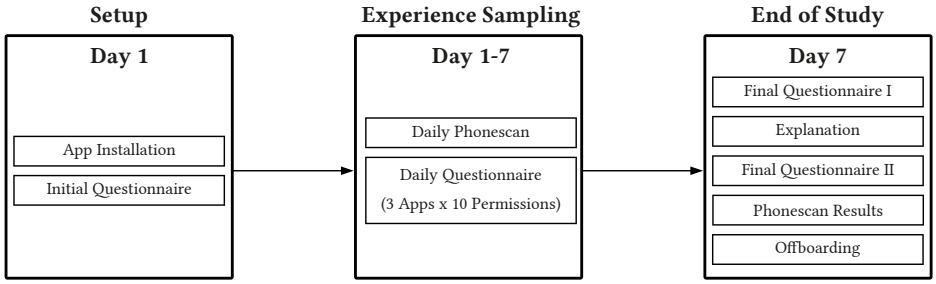


Figure 1: Study Procedure: first, users installed our study application and filled an initial questionnaire on demographics. Next, they completed seven experience sampling questionnaires (one per day, days could be non-consecutive). After completing seven daily questionnaires, participants reached the final part of the study, which included a final questionnaire and an explanation of risky permission sets. We also provided participants with the results of their phone scans and offboarding information.

4. *Final Questionnaire I:* Participants were asked to rate their general awareness of granting permissions at install and runtime (5-point Likert scales).
5. *Explanation & Final Questionnaire II:* We gave participants an explanation on potential privacy and security implications of granting permissions, and on specific dangerous combinations. We then asked them to rate their awareness and understanding of these aspects. Participants could provide additional feedback in a text field.
6. *Results:* Finally, we provided participants with results from their phone scans, in particular regarding potentially dangerous permission combinations. We recommended reviewing such permissions.
7. *Offboarding:* On the last screen in the app, we informed participants that they completed all parts of the study and that they would receive their additional compensation once we verified the completeness of their dataset. We also requested them to uninstall the study app.

## 3.4 Participants & Recruitment

We recruited 110 participants through Prolific [4]. Participants were required to use Android phones, be able to follow the study in English, and be willing to install the study app.

Of these, 107 participants completed the full app setup procedure and, thus, received an initial payment through Prolific. 85 participants completed the whole study duration and final questionnaire. 81 participants passed 2 of 4 attention checks (3 on different days as part of the daily questionnaire, 1 as part of the final questionnaire) and received an additional bonus payment. Note that for analysis, we require participants to pass the attention check in the final questionnaire and at least two attention checks in the dailies (cf Section 3.7), which excluded another 4 participants, resulting in our final number of 77 participants included in the data evaluation. After accessing our study through Prolific, participants were redirected to a university website, where we provided the installation instructions and app download. All further steps were done directly in the app (cf. Section 3.3). We ran the study in January and February 2026 for a total of 18 days.

## 3.5 Ethics Statement

This study received approval from the Institutional Review Board (IRB) at LMU Munich. All participants provided informed consent prior to participation. Participation was voluntary, and participants were informed that they could withdraw at any time without penalty or loss of compensation. Only participants who gave their consent could participate in the study. If consent was revoked during the study, participation was stopped and the data of the respective participants deleted.

The study was designed to involve minimal risk, and no personally identifying information was collected beyond what was necessary for compensation via the Prolific platform. Only the research team had access to the anonymized dataset.

Participants were compensated through Prolific with £1.5 for the setup and an additional payment of £6.5 if they completed all parts of the study, consistent with fair compensation practices for the estimated time commitment (£8 for an estimated time commitment of 50 minutes in total).

While we did not disclose our research interest in risky permission combinations at the start of the study, we made sure to inform participants in the final survey about both, the potential risks associated with certain permission combinations, and the respective risky permission combinations we detected on their devices through the phone scan data. The offboarding contained information about which apps obtained access to which risky permission combinations, and a suggestion to review the respective apps and their permissions.

## 3.6 Limitations

Our work has several limitations. First, we do not explicitly map each permission-based attack vector to a concrete, real-world attack scenario. The primary goal of this study is to examine users' perceptions of the risks associated with combinations of permissions, as well as the prevalence of such combinations in real applications. In addition, typical users generally lack detailed knowledge of how mobile attacks are executed. For these reasons, we did not incorporate an explicit mapping between permission sets and specific attacks into

the experimental design. Future work could address this by systematically linking permission combinations to concrete attack techniques and evaluating user understanding.

Second, our study focuses on the Android operating system. Although Android is the most widely used mobile OS, permission models differ across platforms, and thus may create different permission sets. Future research could extend our analysis by comparing permission sets and user risk perceptions between Android and iOS.

Third, at the application level, our data collection does not capture apps that were installed during the study period. We consider this limitation to have minimal impact on our findings, as we assume that, for most participants, the majority of installed apps were already present prior to the experiment. Moreover, many commonly used applications are preinstalled or installed early in device usage, and thus are likely to be represented in our dataset.

### 3.7 Data Analysis and Cleaning

A total of 85 participants completed all parts of the study, including the final questionnaires. From those participants, we excluded anyone who failed two or more attention checks. This step excluded eight participants, leaving 77 remaining participants we included in our analysis. Our dataset contained phone scan data and dailies for each of the 77. After inspecting the data, we excluded incomplete data points, e.g., due to missing data from the dailies, broken phone scans, or other technical issues. With this step, we excluded specific apps for specific participants, resulting in 151 apps that we excluded across participants. The final dataset consists of the following:

- 77 participants with an initial phone scan, initial questionnaire, and final questionnaire each
- 539 dailies
- 14400 data entries

We used GenAI to help with generating data analysis scripts.

## 4 Results

In the following, we present the results from the automated data logging and user feedback (cf. Section 3.2 for details).

### 4.1 Descriptive Overview

We include a total of 77 participants in our analysis (see Section 3.7 for details on data cleaning).

**Demographics** Participants were 18 to 72 years old ($M = 30.64$, $SD = 10.04$). Of those, 41 identified as male, 34 as female, and 2 as non-binary. Most participants were employed full time ($n = 32$), others were self-employed or students (13 each). A total of 35 participants held a Bachelor's Degree, 19 High School, and 15 Master's. Table 2 provides an overview.

| | | Count | Percent |
|---|---|---|---|
| **Education** | Bachelor's Degree | 35 | 45.5% |
| | High School | 19 | 24.7% |
| | Master's Degree | 15 | 19.5% |
| | Vocational Training | 5 | 6.5% |
| | Doctorate / PhD | 2 | 2.6% |
| | Prefer not to say | 1 | 1.3% |
| **Employment** | Employed full-time | 32 | 41.6% |
| | Self-employed | 13 | 16.9% |
| | Student | 13 | 16.9% |
| | Employed part-time | 6 | 7.8% |
| | Unemployed | 6 | 7.8% |
| | Unable to work | 3 | 3.9% |
| | Retired | 2 | 2.6% |
| | Other | 1 | 1.3% |
| | Prefer not to say | 1 | 1.3% |
| **Gender** | Male | 41 | 53.2% |
| | Female | 34 | 44.2% |
| | Non-binary | 2 | 2.6% |
| **Nationality** | Poland | 8 | 10.4% |
| | Mexico | 6 | 7.8% |
| | South Africa | 6 | 7.8% |
| | Brazil | 6 | 7.8% |
| | Portugal | 6 | 7.8% |
| | Spain | 5 | 6.5% |
| | France | 5 | 6.5% |
| | Italy | 4 | 5.2% |
| | United Kingdom | 4 | 5.2% |
| | Kenya | 3 | 3.9% |
| | Canada | 3 | 3.9% |
| | Estonia | 2 | 2.6% |
| | Germany | 2 | 2.6% |
| | Greece | 2 | 2.6% |
| | India | 2 | 2.6% |
| | Philippines | 2 | 2.6% |
| | Other Countries (1 participant each) | 11 | 14.3% |

Table 2: Demographics: Summary of participants' demographics including reports of education, employment status, gender, and nationality.

**Initial Phone Scan** From the initial phone scan, we found that participants had 170-668 apps installed on their phones ($M = 372.8$). In total, the first phone scans of all participants comprised 28,707 apps (5,486 unique apps). Appendix C, Table 8 provides an overview of the most prominent apps. These apps declared a total of 131,083 Runtime permissions in their manifests (4.6 per app on average). Of those, about half were granted (51%, 66,807 granted Runtime permissions, 2.3 per app on average). Also, these apps requested a total of 205,360 Install-time permissions in their manifests (7.2 per app on average). Figure 2 shows an overview of the Runtime permission states in the initial phone scan.

Permission states for the PermPair permissions as recorded from the first phone scans can be found in Figure 3. Four of the permissions are Install-time permissions and therefore always granted, and the two Runtime permissions `WRITE_EXTERNAL_STORAGE` and `READ_PHONE_STATE` are denied in 62.5% and 53.9% cases, respectively.

**Daily Questionnaires** The dailies (7 per participant) covered 654 unique apps (14 to 21 per participant after the data cleaning, $M = 18.7$, with 3 apps initially questioned per day).

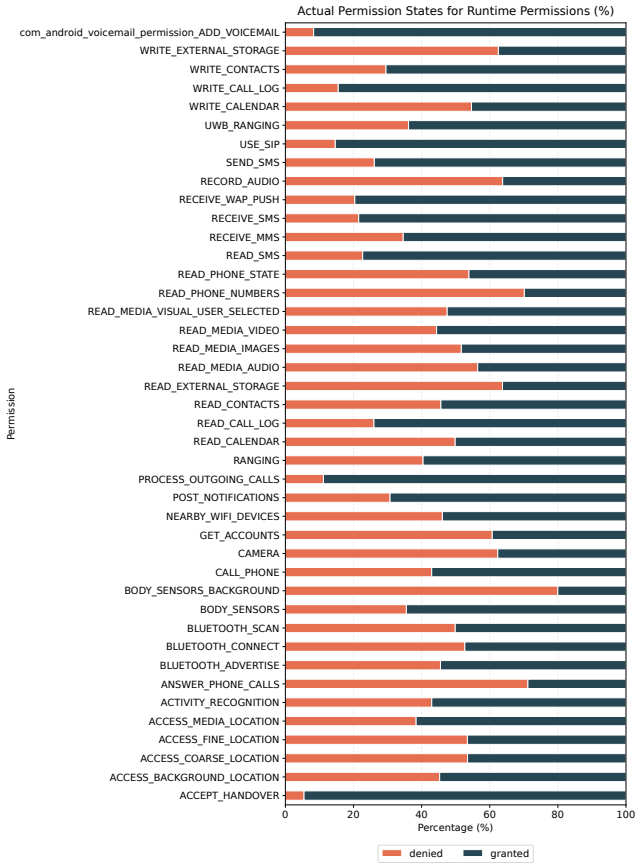


Figure 2: Runtime Permission States: Overview of Runtime permissions present in the first scans across all participants.

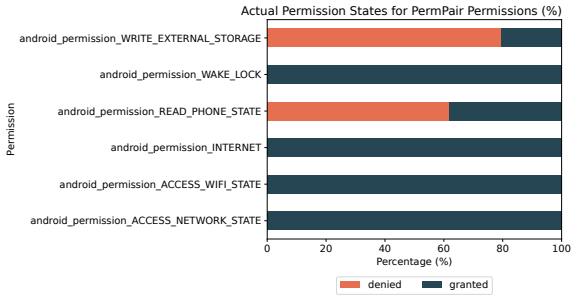


Figure 3: PermPair Permission States: Overview of PermPair permissions that are present in the first phone scans across all participants. Note that four of the permissions are Install-time permissions and, thus, are always granted.

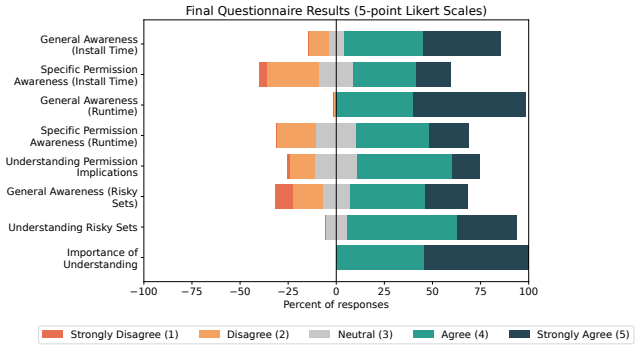


Figure 4: Final Questionnaire Results: Results for Likert items on awareness and understanding of privacy permissions (Install-time, Runtime, Risky Sets) on a 5-point scale.

Apps were balanced across usage categories: 520 *most recently*, 496 *occasionally*, and 450 *least recently*. Table 3 provides an overview of these apps. We include answers for a total of 14,400 permissions in the analysis, 4,364 Runtime, 7,707 Install-time (2,329 other/unclassified), and 6,770 PermPair permissions (note that these comprise both, Runtime and Install-time permissions, cf. Section 3.1).

## 4.2 RQ1: Users' Awareness of Permissions & Implications

In the final questionnaire (cf. Appendix B for the questions and Figure 4 for an overview of the results), participants reported a general awareness of privacy permissions and respective implications. In particular, they stated to be generally rather aware of granting permissions at install-time ($Mdn = 4$, $SD = 0.95$) and runtime ($Mdn = 5$, $SD = 0.57$). More specifically, they stated awareness for specific permissions granted to an app at install-time ($Mdn = 4$, $SD = 1.18$) and runtime ($Mdn = 4$, $SD = 1.04$) along with a general understanding of implications ($Mdn = 4$, $SD = 0.93$). After we provided them with an explanation of *risky permission sets*, participants stated mixed awareness ($Mdn = 4$, $SD = 1.12$) and understanding ($Mdn = 4$, $SD = 0.63$) of these. All participants agreed that understanding permissions is important for them to protect their privacy and security ($Mdn = 5$, $SD = 0.50$).

Note that, with the final questionnaire, we intended to capture participants' self-reported assessments, not necessarily their actual awareness. We approximate the actual awareness by looking at response accuracy, see Section 4.3.

## 4.3 RQ2: Misconceptions

To understand potential misconceptions, we looked at the answers from the dailies and compared them to the current actual state of the specific app's permissions, as extracted from the phone scan on the same day the daily was recorded. In the dailies, participants could answer with *"Yes"*, *"No"*, and *"Don't know"*. Figure 5 shows participants' answers regarding the individual PermPair permissions across all dailies. The actual permission state could only have two values, either *"Granted"* or *"Denied"*. To compare responses to the actual state, we categorized responses as correct when the participant said "Yes" for granted permission and "No" for denied

| App | Count | Usage Category | | |
|---|---|---|---|---|
| | | most | occasionally | least |
| Chrome | 36 | 36 (100.0%) | | |
| Gmail | 31 | 29 (93.5%) | 2 (6.5%) | |
| WhatsApp | 30 | 27 (90.0%) | 2 (6.7%) | 1 (3.3%) |
| Google | 23 | 19 (82.6%) | 4 (17.4%) | |
| Spotify | 23 | 19 (82.6%) | 4 (17.4%) | |
| Instagram | 23 | 21 (91.3%) | 2 (8.7%) | |
| Messages | 20 | 16 (80.0%) | 4 (20.0%) | |
| Live Transcribe & Sound Notifications | 19 | | 3 (15.8%) | 16 (84.2%) |
| YouTube | 19 | 16 (84.2%) | 3 (15.8%) | |
| Google Play Store | 18 | 14 (77.8%) | 4 (22.2%) | |
| Google TV | 17 | | 3 (17.6%) | 14 (82.4%) |
| Android Switch | 17 | | 5 (29.4%) | 12 (70.6%) |
| Meet | 17 | | 6 (35.3%) | 11 (64.7%) |
| Facebook | 17 | 11 (64.7%) | 5 (29.4%) | 1 (5.9%) |
| Security | 15 | 15 (100.0%) | | |
| Telegram | 15 | 7 (46.7%) | 7 (46.7%) | 1 (6.7%) |
| Reminder | 13 | | 1 (7.7%) | 12 (92.3%) |
| Messenger | 12 | 7 (58.3%) | 4 (33.3%) | 1 (8.3%) |
| Photos | 12 | 1 (8.3%) | 11 (91.7%) | |
| Samsung Internet | 12 | 7 (58.3%) | 2 (16.7%) | 3 (25.0%) |
| Outlook | 12 | 7 (58.3%) | 3 (25.0%) | 2 (16.7%) |
| Discord | 12 | 7 (58.3%) | 3 (25.0%) | 2 (16.7%) |

Table 3: Apps Covered in Dailies: Overview of apps that we covered in the daily questionnaires across all participants, along with the usage category. The list is sorted by number of occurrences in the dailies, ties at rank 20 included.

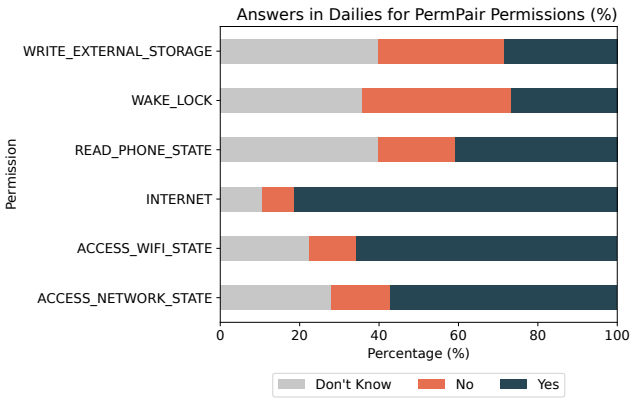


Figure 5: Participants' responses from the dailies, whether they think a certain permission is granted or not, grouped by PermPair Permissions.

permission. If the permission was granted but participants answered "No", this was an incorrect answer; the same applies to denied permissions and "Yes" answers. We looked at "Don't know" answers as a third category to highlight the difference between uncertainty and actual incorrect answers.

**Response Correctness** Table 4 provides an overview of the response correctness, grouped by permission type. For Install-time permissions, answers were correct in 48.81% of cases, while 21.63% were incorrect and 29.56% were "Don't know" answers. For Runtime permissions, 39.30% were correct, 29.22% incorrect, and 31.48% were "Don't know" answers. For PermPair permissions, the majority of answers were correct (52.33%), while 20.25% were incorrect and 29.56% "Don't know" answers. Figure 6 shows the correct/incorrect answers for PermPair permissions specifically.

Table 5 provides an overview of the response correctness, grouped by app usage category. For the most recently used apps, answers were correct in 47.06% of cases, with 24.70% incorrect answers and 28.24 "Don't know" answers. Occasionally used apps had 44.52% correct answers, 25.32% incorrect answers, and 30.16% "Don't know" answers. For the least recently used apps, 37.22% of answers were correct, 25.69% were incorrect, and 37.08% of answers were "Don't know".

**Don't Know Answers** The dailies show that in 31.55% of cases, participants chose the *"Don't know"* option. We see two ways how this could be interpreted: *"Don't know"* as an answer might indicate that participants were not confident in assessing the state of the respective permission. Another possibility is that participants selected this answer when they were not sure what a certain permission meant or to which functionality it referred. When referring to the permissions, we used mostly short names close to the technical identifier (e.g. "Read call log" for "`Manifest.permission.READ_CALL_LOG`"). While this was a practical, unified way to refer to the permissions, the descriptions shown to users on Android might differ. Therefore, participants might be familiar with a specific functionality, but not be able to (correctly) associate it with the respective permission name used in the study. We therefore interpret this answer option as an indicator of the level of uncertainty and lack of knowledge regarding permissions and their state, or unfamiliarity with the permission name that belongs to a specific functionality.

| Permission Type | Correct | Incorrect | Don't Know |
|---|---|---|---|
| PermPair | 52.33% | 20.25% | 27.41% |
| Runtime | 39.30% | 29.22% | 31.48% |
| Install-time | 48.81% | 21.63% | 29.56% |

Table 4: Response Correctness Overview: The percentages indicate whether participants' answer was in line with the actual state of the permission for the different **permission types**. Don't Know answers are considered separately.

| App Usage | Correct | Incorrect | Don't know |
|---|---|---|---|
| most recently | 47.06% | 24.70% | 28.24% |
| occasionally | 44.52% | 25.32% | 30.16% |
| least recently | 37.22% | 25.69% | 37.08% |

Table 5: Response Correctness Overview: The percentages indicate whether participants' answer was in line with the actual state of the permission for the different **app categories**. Don't Know answers are considered separately.

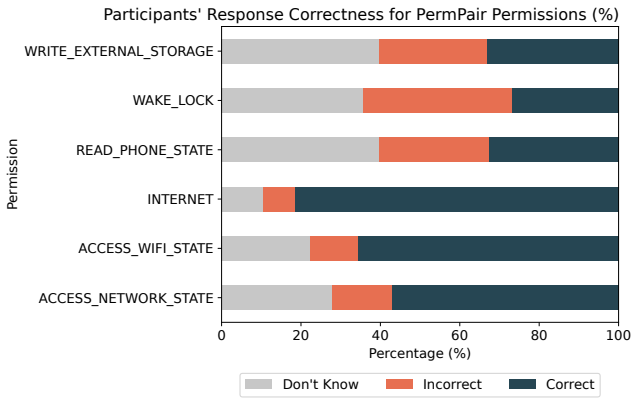


Figure 6: Comparison whether what users reported in the dailies (see Figure 5) aligns with the actual permission state recorded in the phone scan for PermPair Permissions.

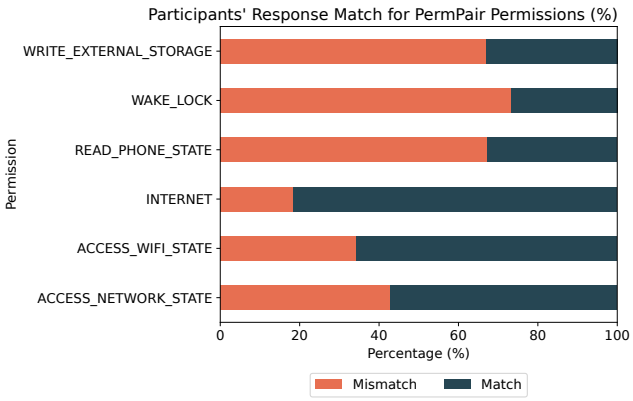


Figure 7: Overview whether user responses match the actual state of a PermPair permission in the phone scan, grouped by "match" (correct answers) and "mismatch" (incorrect and "Don't know" answers).

**False Positives vs. False Negatives** When looking at incorrect answers, we further differentiate between "False Positive" and "False Negative" answers. With "False Positive" we refer to data points where the permission is actually denied, but the participant answered that they think the app has access to the permission. On the other hand we call it a "False Negative" response when the participant thinks a permission is denied when it is actually granted. Note that for Install-time permissions there are no False Positives, as Install-time permissions are always granted. An overview of the False Positive and False Negative responses can be found in Table 6.

| Permission Type | False Positive | False Negative |
|---|---|---|
| PermPair | 4.40% | 15.85% |
| Runtime | 23.95% | 5.27% |
| Install-time | n/a | 21.63% |

Table 6: False Negatives and False Positives for different permission Types. False Negatives are permissions that are granted but the participant answered that they were denied. False Positives are permissions that are denied but the participant answered that they were granted.

**Match vs. Mismatch** For the statistical analysis (see Section 4.4), we further grouped answers into "match" vs. "mismatch" answers, where matching answers refer to correct answers, while mismatched answers include both, incorrect and "Don't know" answers. This allows us to make statements about the true answers, i.e. participants' answer matched the actual state of the permission, but there is no differentiation between a wrong and an uncertain answer. Figure 7 shows the answers grouped by matched and mismatched answers for PermPair permissions.

## 4.4 Factors Related to Response Correctness

We used a generalized linear mixed-effects model (GLMM; binomial family, logit link) to model response correctness [22], defined as whether a participant's report matched the actual permission state (match vs. mismatch, see above). The model included random intercepts for participant, app, and permission to account for the cross-classified repeated-measures structure (multiple responses per participant, per app and per permission). Thus, our model specifications yield valid inference for the fixed effects of permission state, permission type, PermPair status, and app-usage category while appropriately accounting for the possible within-cluster dependencies.

With an intercept of -2.45 (log-odds), the model predicts a baseline probability of a correct response of approximately 0.08 for the reference categories (denied permission, Install-time Permission, non-PermPair, least-recently-used apps) at the mean of the random effects (i.e., for an "average" participant, app, and permission). Fixed-effect coefficients are reported on the log-odds scale; exponentiating these estimates yields odds ratios (ORs), which quantify the multiplicative change in the odds of a correct response relative to the corresponding reference level. All significant fixed effects were positive, indicating higher odds of correctness when comparing each category to its baseline.

The model revealed a significant increase in the likelihood of a correct response (matching the actual permission state) for different factors (Table 7 provides an overview of model estimates): Permission state was a strong predictor of correctness: granted permissions were associated with substantially higher odds of a correct response than denied permissions ($\beta = 1.115$, $SE = 0.07659$, $p < 0.05$, $OR \approx 3.05$), i.e., the odds of responding correctly were about three times higher for granted than for denied permissions. Permission type also mattered, with Runtime permissions showing higher

| | $\beta(log-odds)$ | Std. Error | $p$ Value | Odds Ratio | Baseline |
|---|---|---|---|---|---|
| Intercept | -2.445 | 0.17610 | <2e-16* | | |
| Permission State: Granted | 1.115 | 0.07659 | <2e-16* | $\approx 3.05$ | Permission State: Denied |
| Permission Type: Other | 0.309 | 0.15942 | 0.05229 | $\approx 1.36$ | Permission Type: Installtime |
| Permission Type: Runtime | 1.211 | 0.18450 | 5.33e-11* | $\approx 3.36$ | Permission Type: Installtime |
| Permpair Permissions | 0.859 | 0.30015 | 0.00422* | $\approx 2.36$ | Non-Permpair Permissions |
| App Category: Most Recently Used | 0.371 | 0.07153 | 2.20e-07* | $\approx 1.45$ | App Category: Least Recently Used |
| App Category: Occasionally Used | 0.374 | 0.06559 | 1.17e-08* | $\approx 1.45$ | App Category: Least Recently Used |

Table 7: GLMM Output for Fixed Effects. $* = p < 0.05$

odds of correctness than Install-time permissions ($\beta = 1.211$, $SE = 0.18450$, $p < 0.05$, $OR \approx 3.36$), i.e. the odds of a correct response were more than tripled for Runtime relative to Install-time permissions, whereas permissions of type "other" did not differ reliably from Install-time permissions ($\beta = 0.309$, $SE = 0.159$, $p > 0.05$). $OR \approx 1.36$). Also, PermPair Permissions were associated with higher odds of correctness than non-PermPair Permissions ($\beta = 0.859$, $SE = 0.30015$, $p < 0.05$, $OR \approx 2.36$), corresponding to a little over a two-fold increase in the odds of correctness. Finally, the app-usage category was also related to correctness. Responses concerning most recently used apps ($\beta = 0.371$, $SE = 0.07153$, $p < 0.05$, $OR \approx 1.45$) and occasionally used apps ($\beta = 0.374$, $SE = 0.06559$, $p < 0.05$, $OR \approx 1.45$) were more likely to be correct than responses concerning least recently used apps – roughly a 45% increase in the odds of correctness.

**Summary** These results indicate that participants are 1) 3.05 times more likely to answer correctly for permissions that are **granted** (compared to permissions that are not granted); 2) 3.36 times more likely to answer correctly for **Runtime** Permissions (compared to Install-time permissions); 3) 2.36 times more likely to answer correctly for **PermPair** Permissions compared to non-PermPair Permissions, and 4) 1.45 times more likely to answer correctly for **most recently used** applications and **occasionally used** applications, both compared to least recently used applications.

## 5 Lessons Learned & Future Work

For our study, we deployed *PermWatch*, a tool we previously presented at SOUPS [33]. We configured it in such a way that phone scans and questionnaires came up once per day until data was recorded on 7 days (days could be non-consecutive). We did not employ any event-triggered questionnaires, but rather let participants choose a time frame in which we notified them randomly. In case they followed the notification prompt, a phone scan was conducted and the daily questionnaire came up afterwards. Missed dailies were repeated the next day. While this ensured a certain number of complete questionnaires, the study duration was somewhat unpredictable. In particular, our 77 final participants took 7 to 15 days to complete our study ($M = 8.17$, $SD = 1.58$). We do not expect the difference in study duration to impact our results as the main purpose of running over several days was to distribute effort given the huge number of apps and permissions being covered, rather than investigating time-based effects that are coupled to specific days. An alternative could be to set a maximum duration from the beginning and, e.g., trigger a final questionnaire after 10 days, independent of how many dailies were filled in between. Also, depending on future research questions, the phone scans could be triggered independently and/or more frequently to increase automated data logging while decreasing user participation.

Furthermore, from 110 recruited participants, 107 completed the initial setup, but only 85 reached the final part of the study. 8 participants dropped out after completing the first daily questionnaire and receiving the initial payment. 5 participants dropped out after completing the fourth daily questionnaire, and 2-3 participants each dropped out on the other days of the study. All 85 participants who reached the seventh daily also completed the final questionnaire. Future studies could thus consider shorter time frames and/or other means to keep participants engaged even after they received the first part of the compensation.

For the study, we recruited participants through Prolific [4] (cf. Section 3.4). While this might bias our sample to a more tech-savvy population, this is also beneficial regarding the setup procedure that is necessary to participate (installing an app from a source outside an app store, setting permissions for this app, etc.), which might be a larger hurdle for participants from other target groups. Also, Prolific allowed us to seamlessly compensate participants for the first part, while keeping the bonus payment open to those completing the additional study days. Moreover, a constant communication channel with participants (through Prolific) was useful not only for them to be able to report bugs, but also for us. In particular, we sent one manual reminder to participants in addition to the app notifications. Depending on which recruitment batch participants were in, and on their individual study progress, this reminder reached participants between the second and seventh day of the study. This served two ma-

jor purposes. First, individual phone settings could prevent notifications from being sent as planned; second, additional reminders helped keeping participants engaged.

Lastly, some dangerous Android permissions discussed in related literature and permissions used as attack vectors in recent incidents are not classified as "dangerous" by Android itself. This gap could affect users' perceptions of risk associated with different permissions and should be considered in future work on Android permissions and user awareness.

## 6 Discussion & Design Recommendations

In this section we look at how the different factors which influence users' response correctness can inform the design of privacy interventions with the goal of reducing privacy risks and increasing user understanding of Android permissions. Systems that implement these recommendations can take over detection, prioritization and selecting adequate timing for (re)considering permission settings, while the user stays in the loop to make the final decision.

### 6.1 Reflecting on PermPair Combinations

The PermPair combinations offer a technical perspective on how potentially malicious apps can be identified. We encountered the PermPair permissions in our data set as being present on users' phones. Based on this data, we presented the respective apps and which PermPair combinations they had access to in the debriefing.

In practice, these lists were quite long. Looking at the PermPair permissions, some are related to similar functionality, e.g. `INTERNET`, `ACCESS_NETWORK_STATE`, and `ACCESS_WIFI_STATE`, which makes them likely to appear in combination. If an app has these three permissions, which are all Install-time permissions, they form three PermPair combinations that would show up in the list. While we informed participants that not every app with PermPair combinations is malicious, we prompted them to review the apps in the list. As five of the ten PermPair Permission combinations contained no Runtime permissions, this left our participants with two choices for apps they were uncertain about: either ignore the information we provided about the risky permissions or uninstall the respective apps from their devices. While this would be effective in getting rid of granted PermPair combinations, users might not take this action for a lot of apps. Focusing instead on PermPair combinations that contain at least one Runtime permission, would offer users a more actionable way to react. By declining one (or both) permissions, they could break a risky combination and take meaningful action to protect their privacy.

With the internet-related permissions, it is easy to speculate why a malicious app that is, e.g., intending to steal user data, would need these permissions. However, from a user perspective, looking at internet permissions is not a useful indicator as many benign apps request it as well, and the actual malicious behavior might stem from other permissions when internet connection is mainly needed to transfer the obtained data. Additionally, when the list of apps that have risky permission combinations becomes too long, it is more likely that participants will skim over the warning all together. Thus, excluding, e.g., very frequent permission combinations from the debriefing could help to focus on a concise list which users actually have the capacity to review.

**Recommendation:** To inform users about dangerous permission combinations, we recommend focusing on combinations that include at least one Runtime Permission to provide an easy option to adapt their settings to a less dangerous state.

### 6.2 Unseen Permissions

Users are three times more likely to correctly assess the state of a Runtime permission compared to Install-time permissions. This confirms our expectation, as Runtime permissions are presented more prominently to the user while they are actively using the app. While this result is not surprising, to the best of our knowledge, there is no recent related work reporting empirical data that supports this assumption. While previous literature (e.g., [17, 19, 29]) examined user perceptions of Android permissions, the Android permission system has since evolved, and differences in user perceptions of Install-time and Runtime permissions are not covered. While Android assigns a protection level [2] to each permission and classifies some as "dangerous" (e.g., Runtime permissions), constructs such as dangerous permission combinations highlight that assigning a protection level to individual permissions can miss certain attack vectors. Therefore, it is important for users to also be aware of other permissions, such as Install-time permissions, to gain an overall understanding of their apps' capabilities. The process of granting permissions simply by installing an app should thus be less hidden and become an active step in the installation process.

**Recommendation:** Install-time permissions are way less prominent as compared to Runtime permissions, i.e., they are only shown once in the app store *before* installation. To better support users staying aware of these unseen permissions, we call providers and developers to a) make these more prominent at the beginning of an installation process, and b) provide an option to review these permissions also *after* installation.

### 6.3 Forgotten Apps

As users' answers were significantly less correct for their least recently used apps (compared to their most recently and occasionally used apps), this presents a need for intervention. Besides reminding users of which rarely used apps they have installed on their device, the fact that these apps are rarely or

no longer used at all opens up an additional opportunity: Users can be prompted to consider deleting apps in this group if they have unwanted permissions granted. While they might still not do this for every app, our assumption is that it is more likely than with other apps still in use. Android already has a feature called "*auto revoke*" [1] with a similar functionality: It takes into account when an app was last used, and automatically revokes Runtime permissions if the last used timestamp passes a certain threshold. Extending this approach by prompting users to uninstall apps after even more time has passed since their last use could help effectively remove unused apps and respective Install-time permissions from users' devices.

**Recommendation:** Apps identified as *"rarely used"* are candidates that users might actually want to delete if they have access to risky permission combinations. Considering usage time or when an app was last used can serve as an indicator of what should be suggested to the user.

### 6.4 Meet Users Where They Are

We found various misconceptions in our daily questionnaires, mainly a) False Positives (e.g., User A thinks permission X is granted, but it is actually denied) and b) False Negatives (e.g., User B thinks permission Y is denied, but it is actually granted). These lead to different levels of privacy risks for these users. While wrongly thinking a permission is granted reveals a misconception worth addressing, e.g., by educating the user and informing them of the actual permission state, the risk associated with wrongly assuming a permission is not granted is higher. Similarly, answers that indicate uncertainty about a permission state reveal that the user is not confident in assessing the permission state on their phone. Depending on the actual permission state, an intervention is more or less urgent, for granted or denied permissions, respectively.

Depending on the user's self-reported awareness regarding permission states, the risk might be amplified if the perception (as captured by the self-reported assessment in the final questionnaire) deviates from the actual accuracy of the responses (approximated by looking at the response accuracy, see Section 4.3). Someone who knows they lack awareness of their granted permissions might be more cautious than someone who mistakenly believes their assessment is accurate. Giving users the opportunity to understand their actual permission states and reflect on their awareness levels could help in mitigating the risks associated with the mismatch between self-reported awareness and answer accuracy.

**Recommendation:** Depending on users' misconceptions, different intervention strategies are suitable. Misconceptions about granted permissions should be addressed immediately, while those about denied permissions are less time-critical and can be addressed at a later, opportune moment.

## 7 Conclusion

In this paper, we present the results from an in-the-wild study ($N = 77$) exploring users' awareness of Android permission states and investigating mismatches between users' perceptions and actual permission settings.

We found various factors that are related to users' response correctness, including permission state, permission type, and app usage category. We propose different ways to address these factors and increase users' awareness and understanding of granted permissions. Depending on the type of incorrectness, the need for an intervention is different: False Positive responses indicate a need for additional information to correct the user's knowledge of permission state, while False Negatives require immediate action to let the user know which permission is actually granted and therefore which functionalities an app has access to. In the context of dangerous permission combinations identified by prior work, we suggest to focus on those that allow for users to take action.

We hope that our results will inform future work designing and evaluating intervention interfaces for privacy permission management. Enabling users to take action is an important step towards protecting their privacy and helping them navigate Android permissions in the long run.

## Acknowledgments

We thank the reviewers for their valuable feedback, Gabriel Knoll for his support with implementing the study app, and Malin Benjes for her input regarding statistics. This research received funding from the German Research Foundation (DFG) under grant agreement no. 521584224, and the Israeli Ministry of Innovation, Science, and Technology.

## References

[1] Android permission controller – auto-revoke for unused apps. https://source.android.com/docs/core/ota/modular-system/permissioncontroller#auto-revoke. last accessed: 2026-02-18.

[2] R API Reference: R.attr.protectionlevel. https://developer.android.com/reference/android/R.attr#protectionLevel. last accessed: 2026-02-18.

[3] Permissions on Android. https://developer.android.com/guide/topics/permissions/overview, 2022. last accessed: 2026-02-18.

[4] Prolific. A Higher Standard of Online Research. https://prolific.co/, 2022. last accessed: 2026-02-18.

[5] Request Runtime Permissions. https://developer.android.com/training/permissions/requesting, 2022. last accessed: 2026-02-18.

[6] Melvin Abraham, Mark McGill, and Mohamed Khamis. What You Experience is What We Collect: User Experience Based Fine-Grained Permissions for Everyday Augmented Reality. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI '24)*. ACM, 2024. doi:10.1145/3613904.3642668.

[7] Nourah Alshomrani, Steven Furnell, and Ying He. Assessing User Understanding, Perception and Behaviour with Privacy and Permission Settings. In *HCI for Cybersecurity, Privacy and Trust: 5th International Conference, HCI-CPT 2023, Held as Part of the 25th HCI International Conference, HCII 2023, Copenhagen, Denmark, July 23–28, 2023, Proceedings*, pages 557–575, 2023. doi:10.1007/978-3-031-35822-7_36.

[8] Ashwaq Alsoubai, Reza Ghaiumy Anaraky, Yao Li, Xinru Page, Bart Knijnenburg, and Pamela J. Wisniewski. Permission vs. App Limiters: Profiling Smartphone Users to Understand Differing Strategies for Mobile Privacy Management. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, 2022. doi:10.1145/3491102.3517652.

[9] Anshul Arora, Sateesh K. Peddoju, and Mauro Conti. PermPair: Android Malware Detection Using Permission Pairs. *IEEE Transactions on Information Forensics and Security*, 15:1968–1982, 2020. doi:10.1109/TIFS.2019.2950134.

[10] Harel Berger, Chen Hajaj, Enrico Mariconti, and Amit Dvir. Crystal Ball: From Innovative Attacks to Attack Effectiveness Classifier. *IEEE Access*, 10:1317–1333, 2022. doi:10.1109/ACCESS.2021.3138628.

[11] Paolo Calciati, Konstantin Kuznetsov, Alessandra Gorla, and Andreas Zeller. Automatically Granted Permissions in Android apps: An Empirical Study on their Prevalence and on the Potential Threats for Privacy. In *Proceedings of the 17th International Conference on Mining Software Repositories*, 2020. doi:10.1145/3379597.3387469.

[12] Michalis Diamantaris, Elias P. Papadopoulos, Evangelos P. Markatos, Sotiris Ioannidis, and Jason Polakis. REAPER: Real-Time App Analysis for Augmenting the Android Permission System. In *Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy*, 2019. doi:10.1145/3292006.3300027.

[13] Verena Distler, Matthias Fassl, Hana Habib, Katharina Krombholz, Gabriele Lenzini, Carine Lallemand, Lorrie Faith Cranor, and Vincent Koenig. A Systematic Literature Review of Empirical Methods and Risk Representation in Usable Privacy and Security Research. *ACM Trans. Comput.-Hum. Interact.*, 28(6), December 2021. doi:10.1145/3469845.

[14] Bharathi Donku, Shahriar Rahman Khan, Tariqul Islam, and Raiful Hasan. Discrepancies in Mobile App Permissions: Exploring Transparency and User Awareness in the Android Ecosystem. In *Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems*, 2025. doi:10.1145/3706599.3719902.

[15] Janna Lynn Dupree, Richard Devries, Daniel M. Berry, and Edward Lank. Privacy Personas: Clustering Users via Attitudes and Behaviors toward Security Practices. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, page 5228–5239, 2016. doi:10.1145/2858036.2858214.

[16] Yusra Elbitar, Alexander Hart, and Sven Bugiel. The Power of Words: A Comprehensive Analysis of Rationales and Their Effects on Users' Permission Decisions. In *Proceedings of the 32nd Network and Distributed System Security Symposium (NDSS)*. Internet Society, 2025. doi:10.14722/ndss.2025.230544.

[17] Adrienne Porter Felt, Elizabeth Ha, Serge Egelman, Ariel Haney, Erika Chin, and David Wagner. Android Permissions: User Attention, Comprehension, and Behavior. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, SOUPS '12, New York, NY, USA, 2012. Association for Computing Machinery. doi:10.1145/2335356.2335360.

[18] Yanick Fratantonio, Chenxiong Qian, Simon P Chung, and Wenke Lee. Cloak and Dagger: From Two Permissions to Complete Control of the UI Feedback Loop. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 1041–1057. IEEE, 2017. doi:10.1109/SP.2017.39.

[19] Alisa Frik, Juliann Kim, Joshua Rafael Sanchez, and Joanne Ma. Users' Expectations About and Use of Smartphone Privacy and Security Settings. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems*, 2022. doi:10.1145/3491102.3517504.

[20] Nina Gerber, Paul Gerber, and Melanie Volkamer. Explaining the privacy paradox: A systematic review of literature investigating privacy attitude and behavior. *Computers & Security*, 77:226–261, 2018. doi:10.1016/j.cose.2018.04.002.

[21] Alejandro Guerra-Manzanares, Hayretdin Bahsi, and Marcin Luckner. Leveraging the first line of defense: a study on the evolution and usage of android security permissions for enhanced android malware detection. *Journal of Computer Virology and Hacking Techniques*, 19(1):65–96, 2023. doi:10.1007/s11416-022-00432-3.

[22] Maurits Kaptein. Using Generalized Linear (Mixed) Models in HCI. In Judy Robertson and Maurits Kaptein, editors, *Modern Statistical Methods for HCI*, pages 251–274. Springer International Publishing, Cham, 2016. doi:10.1007/978-3-319-26633-6_11.

[23] Shahriar Rahman Khan and Raiful Hasan. 'Watch for the Consequences': Visualizing Permission Risks to Improve Privacy Decisions in Android. In *Adjunct Proceedings of the Twenty-First Symposium on Usable Privacy and Security*. USENIX Association, 2025. URL: https://www.usenix.org/conference/soups2025/presentation/khan-poster.

[24] Sarah Prange, Pascal Knierim, Gabriel Knoll, Felix Dietz, Alexander De Luca, and Florian Alt. "I do (not) need that Feature!" – Understanding Users' Awareness and Control of Privacy Permissions on Android Smartphones. In *Twentieth Symposium on Usable Privacy and Security (SOUPS 2024)*, pages 453–472, Philadelphia, PA, August 2024. USENIX Association. URL: https://www.usenix.org/conference/soups2024/presentation/prange.

[25] Muhammad Sajidur Rahman, Pirouz Naghavi, Blas Kojusner, Sadia Afroz, Byron Williams, Sara Rampazzi, and Vincent Bindschaedler. PermPress: Machine Learning-Based Pipeline to Evaluate Permissions in App Privacy Policies. *IEEE Access*, 10:89248–89269, 2022. doi:10.1109/ACCESS.2022.3199882.

[26] Tejpal Sharma and Dhavleesh Rattan. Characterization of android malwares and their families. *ACM Comput. Surv.*, 57(5), January 2025. doi:10.1145/3708500.

[27] Yash Sharma and Anshul Arora. A comprehensive review on permissions-based Android malware detection. *International Journal of Information Security*, 2024. doi:10.1007/s10207-024-00822-2.

[28] Yash Sharma and Anshul Arora. IPAnalyzer: A novel Android malware detection system using ranked Intents and Permissions. *Multimedia Tools and Applications*, pages 1–52, 2024. doi:10.1007/s11042-024-18511-6.

[29] Bingyu Shen, Lili Wei, Chengcheng Xiang, Yudong Wu, Mingyao Shen, Yuanyuan Zhou, and Xinxin Jin. Can Systems Explain Permissions Better? Understanding Users' Misperceptions under Smartphone Runtime Permission Model. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 751–768, 2021.

[30] Güliz Seray Tuncay, Soteris Demetriou, Karan Ganju, and Carl A Gunter. Resolving the Predicament of Android Custom Permissions. In *Network and Distributed Systems Security (NDSS) Symposium*, 2018. doi:10.14722/ndss.2018.23210.

[31] Yang Wang, Jun Zheng, Chen Sun, and Srinivas Mukkamala. Quantitative Security Risk Assessment of Android Permissions and Applications. In *Data and Applications Security and Privacy XXVII: 27th Annual IFIP WG 11.3 Conference, DBSec 2013, Newark, NJ, USA, July 15-17, 2013. Proceedings 27*, pages 226–241. Springer, 2013. doi:10.1007/978-3-642-39256-6_15.

[32] Primal Wijesekera, Arjun Baokar, Ashkan Hosseini, Serge Egelman, David Wagner, and Konstantin Beznosov. Android Permissions Remystified: A Field Study on Contextual Integrity. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 499–514, Washington, D.C., August 2015. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/wijesekera.

[33] Verena Winterhalter, Anouk Moreno, Sarah Prange, Harel Berger, and Florian Alt. PermWatch: A Tool for In-Situ Research on Users' Awareness and Control of Android Permissions. In *Adjunct Proceedings of the Twenty-First Symposium on Usable Privacy and Security*, SOUPS '25. USENIX Association, 2025. URL: https://www.usenix.org/system/files/soups2025_poster26_abstract-winterhalter-permwatch.pdf.

[34] Yuxi Wu, Jacob Logas, Devansh Ponda, Julia Haines, Jiaming Li, Jeffrey Nichols, W. Keith Edwards, and Sauvik Das. Modeling End-User Affective Discomfort With Mobile App Permissions Across Physical Contexts. In *Proceedings of the 12th Symposium on Usable Security and Privacy (USEC)*, San Diego, CA, 2025. Internet Society / NDSS Program. doi:10.14722/usec.2025.23023.

[35] Jin Zhang, Chennan Zhang, Xiangyu Liu, Yuncheng Wang, Wenrui Diao, and Shanqing Guo. ShadowDroid: Practical Black-box Attack against ML-based Android Malware Detection. In *2021 IEEE 27th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 629–636. IEEE, 2021. doi:10.1109/ICPADS53394.2021.00084.

[36] Shikun Zhang, Lily Klucinec, Kyerra Norton, Norman Sadeh, and Lorrie Faith Cranor. Exploring Expandable-Grid Designs to Make iOS App Privacy Labels More Usable. In *Twentieth Symposium on Usable Privacy and Security (SOUPS 2024)*, pages 139–157, Philadelphia, PA, August 2024. USENIX Association. URL: https://www.usenix.org/conference/soups2024/presentation/zhang.

# A Study Protocol

## A.1 Installation Guide

We ask you to download and install the ***PermWatch*** app via the APK-File provided above. As such, the following steps will be necessary:

1. Download the APK file to your phone
2. navigate to your download folder
3. click/touch on the APK file you just downloaded
4. confirm that you want to install the app
5. Note: a warning might appear as you install this application outside from the regular Playstore. Please click on ***"More Details"***, then ***"Install without scanning"***. The app will then be installed on your phone.
6. For the notifications to show up properly over the entire duration of the survey, it might be necessary to **remove battery restrictions** in the settings of your phone. To do so, navigate to the "Apps" list in your phone's settings. Select the ***PermWatch*** App, tap on "Battery" and select "Unrestricted" (The exact location of this setting might vary depending on your Phone or Android Version).
7. Navigate to your app menu and start the ***PermWatch*** app.

The app will now guide you through the setup and initial questionnaire.Throughout the setup process, please make sure to allow the ***"Usage Access"*** permission, as this is necessary for us to collect information on app permissions. Note: **the collected data will not be linked to your identity!**

Please confirm that you have **downloaded and installed** the app before you continue. You can not return to this page!

- I have downloaded the APK file and installed the app. I want to continue.

Please follow the instructions on your phone to complete the app setup and first daily questionnaire. You will then receive the completion code in the app. Please enter it manually to Prolific after completing the survey.

Please confirm here once you completed the first part of the study in the app

- I completed setup and daily questionnaire and see the completion code in the app.

If you want to receive the bonus payment at the end of the study, please make sure to complete all 7 dailies. If the notification on a given day does not show up, you can still open the app manually and complete the daily.

## A.2 Initial Questionnaire

In the initial questionnaire, we asked for the following information:

1. Prolific ID (text entry)
2. Age (slider with numbers)
3. Gender (select from dropdown menu)
4. Country of Nationality (select from dropdown menu)
5. Education Level (select from dropdown menu)
6. Employment Status (select from dropdown menu)

Once participants provided their demographic information, the first phone scan was started.

## A.3 Daily Questionnaire

In the daily questionnaire, we presented 3 apps to the participant in the following way:

- Which of these permissions does the app currently have access to?
- Name app 1 & Icon app 1
    1. Name Permission 1;
       Answer options: Yes / No / I don't know
    2. Name Permission 2;
       Answer options: Yes / No / I don't know
    3. Name Permission 3;
       Answer options: Yes / No / I don't know
    4. ...
    5. Name Permission 10;
       Answer options: Yes / No / I don't know

This questionnaire was repeated for the 3 apps that were selected for the daily.

# B Final Questionnaire

On the first page of the questionnaire, we asked (on 5-point Likert Scales):

1. I am generally aware that I grant permissions by installing an app. (*General Awareness (Installtime)*)
2. I am aware of the specific permissions I grant during installation. (*Specific Permission Awareness (Installtime)*)
3. I am generally aware that apps can request permissions in the form of pop-ups while being used. (*General Awareness (Runtime)*)

4. I am aware of the specific permissions I grant while using an app. (*Specific Permission Awareness (Runtime)*)

5. I generally understand the implications of granting particular permissions. (*Understanding Permission Implications*)

We then provided participants with the following explanation:

> Granting individual permissions can impact the security and privacy of the phone by giving an app access to device functionalities and user data. Granting certain combinations of permissions can further increase that risk. We call these combinations *risky permission sets*.

On the next page of the questionnaire, where the explanation was still available, we asked (on 5-point Likert Scales):

1. I was aware of the existence of risky permission sets. (*General Awareness (Risky Sets)*)

2. I understand why some sets of permissions can be risky. (*Understanding Risky Sets*)

3. Understanding app permissions is important for protecting my privacy and security. (*Importance of Understanding*)

The last question was an open text field, asking for any final feedback: "Do you have any comments to share about your experience so far?"

# C Additional Data

## C.1 Initial Phone Scan

| App (package name) | Count |
|---|---|
| com_facebook_appmanager | 68 |
| com_facebook_services | 68 |
| com_facebook_system | 67 |
| com_whatsapp | 66 |
| com_google_ar_core | 54 |
| com_facebook_katana | 49 |
| com_spotify_music | 49 |
| com_microsoft_skydrive | 43 |
| com_discord | 40 |
| com_facebook_orca | 40 |
| org_telegram_messenger | 40 |
| com_microsoft_office_outlook | 39 |
| com_sec_spp_push | 38 |
| com_microsoft_appmanager | 38 |
| com_sec_bcservice | 36 |
| com_sec_imslogger | 36 |
| com_sec_imsservice | 36 |
| com_openai_chatgpt | 36 |
| com_hiya_star | 36 |
| com_sec_epdg | 36 |
| com_sec_epdgtestapp | 36 |
| com_google_audio_hearing_visualization_accessibility_scribe | 36 |
| com_sec_phone | 36 |
| com_sec_sve | 36 |
| com_sec_unifiedwfc | 36 |
| com_sec_usbsettings | 36 |
| com_sec_vsim_ericssonnsds_webapp | 36 |
| com_osp_app_signin | 36 |
| com_samsung_app_newtrim | 36 |
| com_samsung_SMT | 36 |
| com_samsung_aasaservice | 36 |
| com_samsung_accessibility | 36 |
| com_wsomacp | 36 |
| com_samsung_advp_imssettings | 36 |
| com_samsung_cmh | 36 |
| com_samsung_safetyinformation | 36 |
| com_samsung_storyservice | 36 |
| com_sec_app_RilErrorNotifier | 36 |

Table 8: Results from the Initial Phone Scan: Overview of the most installed applications that are not including "android" in their package name (we also exclude *PermWatch* from this list). We cut off the list at the 20th rank (*Count* = 36), and include all further packages with this count.

## C.2 Daily Surveys

The last part of the Appendix contains figures for the daily answers, response correctness, and response match for the different permission types, respectively. For Runtime permissions see Figures 8, 10 and 12, for Install-time permissions see Figures 9, 11 and 13.

### C.2.1 Daily Answers

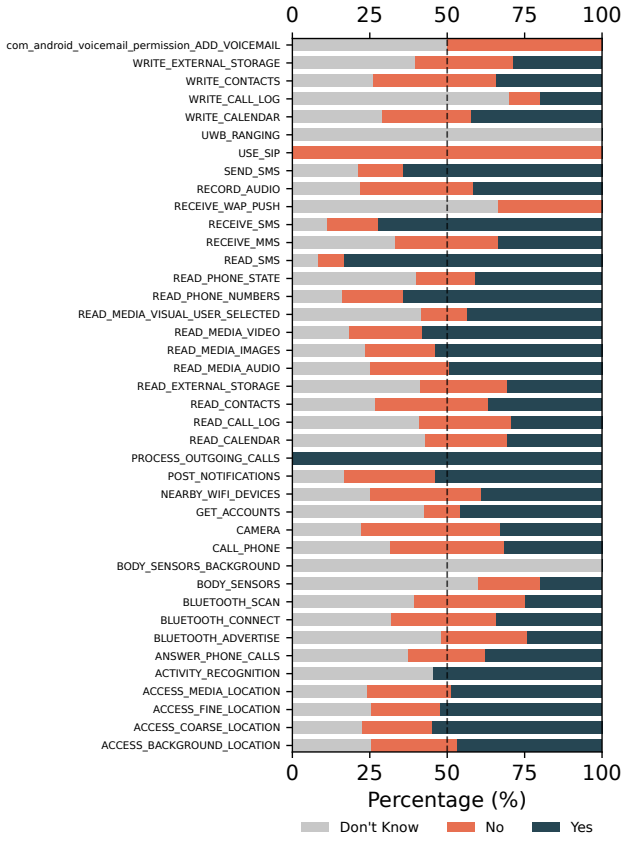


Figure 8: Participants' responses from the dailies whether they think a certain **Runtime** Permission is granted or not.

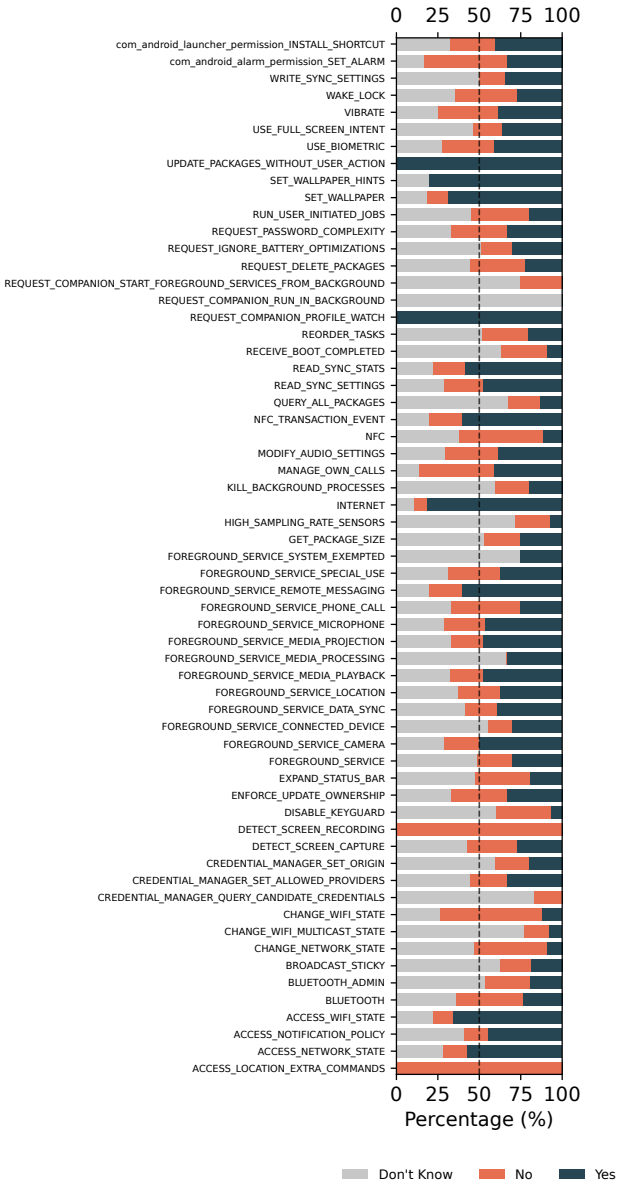


Figure 9: Participants' responses from the dailies whether they think a certain **Install-time** Permission is granted or not.

### C.2.2 Response Correctness

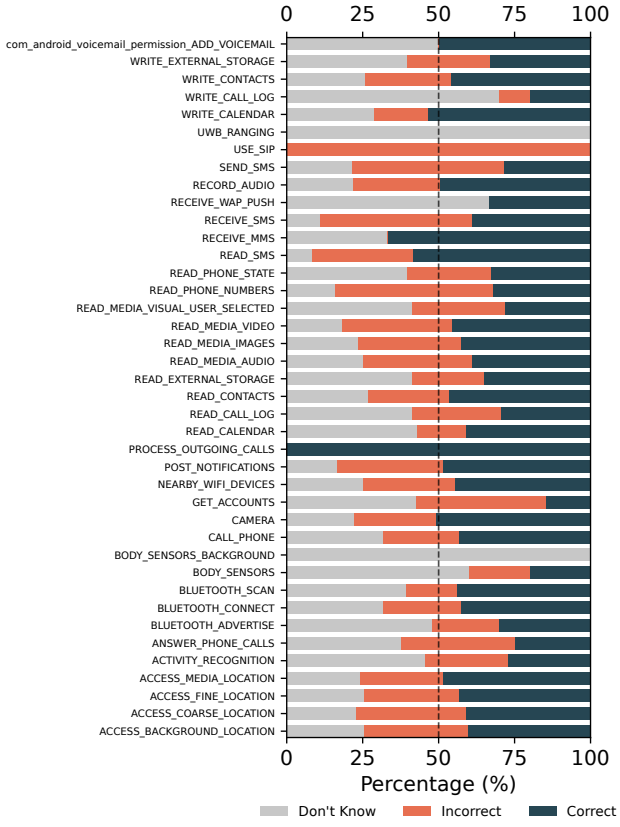


Figure 10: Comparison whether what users reported in the Dailies (see Figure 8) aligns with the actual **Runtime Permission** state recorded in the respective phone scan.

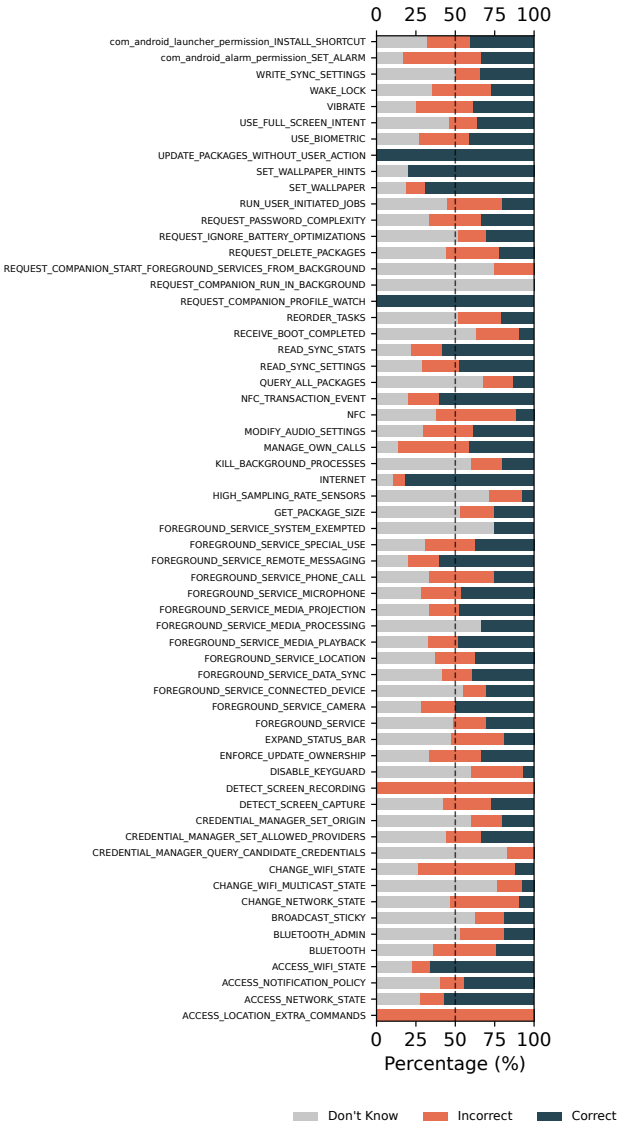


Figure 11: Comparison whether what users reported in the Dailies (see Figure 9) aligns with the actual **Install-time permissions** state recorded in the phone scan.

### C.2.3 Response Match

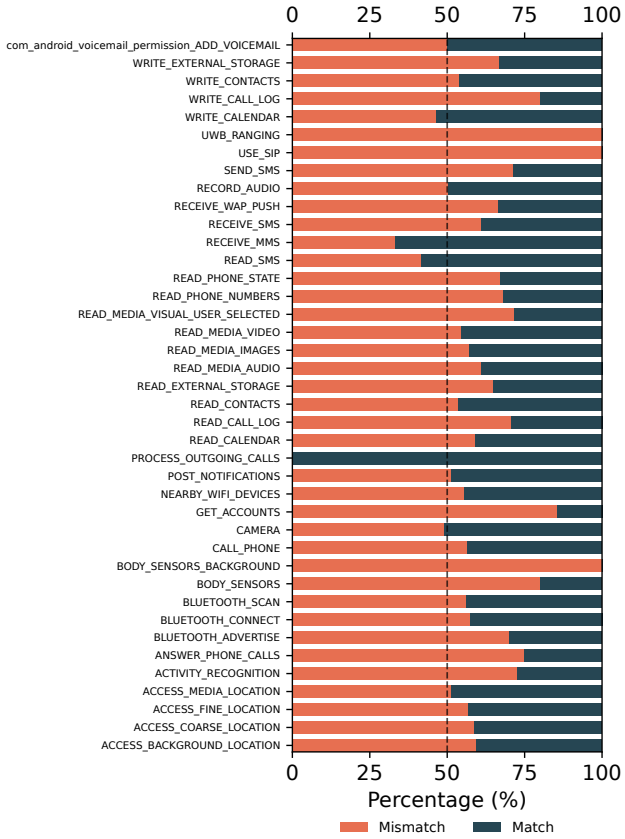


Figure 12: Overview whether user responses for **Runtime permissions** match the actual state of the phone scan, grouped by "match" (correct answers) and "mismatch" (incorrect and "Don't know" answers).

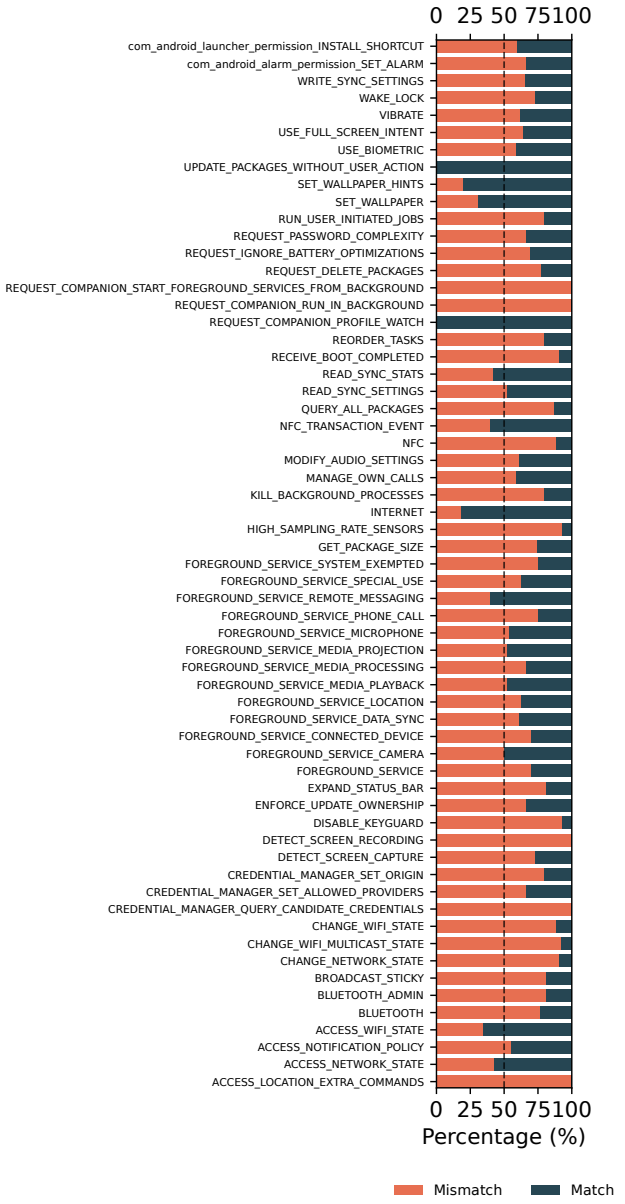


Figure 13: Overview whether user responses for **Install-time permissions** match the actual state of the phone scan, grouped by "match" (correct answers) and "mismatch" (incorrect and "Don't know" answers).