Effect of User Expectation on Mobile App Privacy: A Field Study

Ricardo Mendes
CISUC
Dept. of Informatics Engineering
University of Coimbra
Coimbra, Portugal
rscmendes@dei.uc.pt

André Brandão, João P. Vilela
CRACS/INESCTEC and CISUC
Dept. of Computer Science
Faculty of Sciences
University of Porto
Porto, Portugal
{up201908477, jvilela}@fc.up.pt

Alastair R. Beresford
Dept. of Computer Science and Technology
University of Cambridge
Cambridge, England
arb33@cam.ac.uk

***Abstract*—Runtime permission managers for mobile devices allow requests to be performed at the time in which permissions are required, thus enabling the user to grant/deny requests in context according to their expectations. However, in order to avoid cognitive overload, second and subsequent requests are usually automatically granted without user intervention/awareness. This paper explores whether these automated decisions fit user expectations. We performed a field study with 93 participants to collect their privacy decisions, the surrounding context and whether each request was expected. The collected 65261 permission decisions revealed a strong misalignment between apps' practices and expectation as almost half of requests are unexpected by users. This ratio strongly varies with the requested permission, the category and visibility of the requesting application and the user itself; that is, expectation is subjective to each individual. Moreover, privacy decisions are most strongly correlated with user expectation, but such correlation is also highly personal. Finally, Android's default permission manager would have violated the privacy of our participants 15% of the time.**

***Index Terms*—Permission Managers, Privacy as Expectations, Contextual Integrity, Mobile Devices, Android**

I. Introduction

The pervasiveness of smart devices and the always connected paradigm have fostered applications that benefit from sensing the environment to provide contextualized services to users. However, this constant collection and flow of information present severe privacy and security risks [1], such as the possibility of disclosure through data breaches.

Due to their inherent capacity to collect high quantities of sensible data, smartphones have implemented permission managers to give users control over which applications can access certain device resources, including sensors and data. Both Android's and iOS's permission managers implement runtime permissions, where an application attempting to access a sensitive resource must first check if it has permission; if not, permission is requested from the user. By using runtime permissions, the permission requests made to the user are well contextualized by the app, typically at the time the resource is required, thus helping users with their decision to allow or deny the request [2], [3]. The problem with this model however, is that, after being allowed once, the permission is typically automatically granted on all subsequent occasions, including when the user is unaware that the app is running [4], [5], thus violating privacy's contextual integrity [6], or in other words, defying users' expectations.

Privacy as contextual integrity is a model that binds privacy to the appropriateness of gathering and disseminating data at each specific context [6]. In this model, context is not limited to time and location, but is instead an abstract sphere that describes a situation and thus can encompass the activity being performed, the roles and norms binding each involved entity, the cultural and political ecosystem, and any other information that characterizes the current status. In this regard, any given data practice might be both appropriate or a violation of privacy depending on the context and on the expectations of the user within that context [6]. In mobile devices, the expectation of a user is their mental model that describes the functionality of an app [7], i.e., what the app does and how it works.

Expectation is important: if an app fully behaves as expected by the user then fewer privacy problems would arise [7]. However, users' expectations and app practices often diverge due to the lack of knowledge by the user [2], [7] or by apps' intrusive practices [4], [5]. Expectations should guide app design and support privacy-aware decisions [2], [7].

This work aims to evaluate the importance of the expectation in privacy decisions through the lens of privacy's contextual integrity [8]. Specifically, to measure the importance of user expectation in privacy decisions and how strongly it varies with changes in the context, such as the visibility of the requesting app and the location of the user. Towards this end, we have collected permission decisions and the respective context and whether the requests were expected (which we simply refer to as expectation) from 93 participants,

This work is supported by project COP-MODE, that has received funding from the European Union's Horizon 2020 research and innovation programme under the NGI TRUST grant agreement no 825618, and the project SNOB-5G with Nr. 045929 (CENTRO-01-0247-FEDER-045929) supported by the European Regional Development Fund (FEDER), through the Regional Operational Programme of Centre (CENTRO 2020) of the Portugal 2020 framework and FCT under the MIT Portugal Program. Ricardo Mendes wishes to acknowledge the Portuguese funding institution FCT - Foundation for Science and Technology for supporting his research under the Ph.D. grant SFRH/BD/128599/2017.

who carried an Android phone with a modified permission manager for at least one week. This work makes the following contributions:

- To the best of our knowledge, this is the first field study to capture the expectation of users regarding runtime permissions at scale and in-situ, thus avoiding potentially aspirational responses that might not align with behavior [9]. We make this dataset available to interested researchers (see Section III-B).
- We uncover a strong misalignment between app practices and the expectation of users. Specifically from the collected data, almost half of requests are unexpected by users, a ratio that mostly varies with the requested permission, category of the requesting app, the visibility of the requesting app and, more importantly, the user.
- Privacy decisions see the strongest correlation with expectation, mainly due to the fact that 90% of expected requests are allowed by users. However, expectation greatly varies with each individual. Thus we conclude that not only is expectation personal but so is the importance of it in privacy decisions.
- Finally, our data shows that Android 9 default permission manager would have resulted in privacy violations 15% of the time, i.e. allowed permission requests that were explicitly denied by our participants when using our permission manager. This behavior is a clear violation of privacy, which potentially originates from the need to improve usability; that is, to reduce the number of requests and therefore avoid warning fatigue and habituation. In fact, with the default permission manager of Android 9, to reach 15% of privacy violations the user would still have to reply to a median of 64 permission request prompts during the 7 day campaign period.

The remainder of this work is structured as follows. Section II presents the related work and Section III describes the methodology of the campaigns and details the data collection tool. Section IV does an exploratory data analysis on the collected dataset, with a focus on privacy decisions and the respective expectation. Section VI concludes this work.

II. Related Work

With runtime permissions, applications must request permission the first time they require access to a sensitive resource, thus allowing finer-grained control over each particular permission for any installed app [2]. By prompting at runtime, permission requests are contextualized by the needs of the application, therefore helping users to make an informed decision [2], [3]. Additionally, it allows for the inclusion of developer explanations in the prompts, clarifying their necessity [10], [11].

The major problem with the current runtime permission model lies not in the permission prompts, but in the resource accesses that are made without user knowledge [4], [5], [12], a clear violation of contextual integrity. After being granted once, apps generally have access to a resource until the user denies it through phone settings, which they typically do not change [3] or, in newer Android versions (from Android 11) until it is automatically reverted to the denied state after a few months of not using the app. In fact, users feel their personal space violated when confronted with apps' intrusive practices [4], [13]. Consequently, industry permission managers found in Android (and iOS) still fail to convey the privacy risks that arise from allowing these permissions [2], [14].

Towards improving privacy awareness, researchers have proposed using personal examples to better convey permission risks [15], crowdsourcing the feelings of uneasiness regarding apps' practices [7], and designing better permission warnings [16] or indicators of resource usage [17]. These types of warnings can show information on how, how often and even for which purpose permissions are used in a non-intrusive approach [18], [19], thus leading users towards knowledgeable privacy decisions. However, the challenges here relate to the decision on which information is relevant and how to clearly present this information such that users understand and act upon these warnings [14], [18]. Furthermore, it has been reported that these notices might "annoy" users when prompted at inconvenient times [4], [20]. Solutions to this latter problem include configurable periodic nudges [4], [21], and contextualized notices [18], where the user is presented with nudges that are relevant to the current context, such as upon occurrence of a specific data practice.

To mitigate warning fatigue and habituation, a state where users become desensitized and therefore promptly dismiss notices, solutions for automatically setting permissions have been proposed. These approaches leverage the correlation between the user privacy decision, the category of the requesting app, requested permission [2], personal preferences [20], [22] and contextual features [23]–[25].

Automated and personalized proposals generally rely on privacy profiles to capture personal preferences, where each profile is a representation of privacy behavior from like-minded individuals [22]. A small of number of profiles is sufficient to describe the diverse space of permission decisions [22] and a profile can be readily assigned through a few questions [20], therefore bootstrapping the automation while minimizing the required input. However, current privacy profile approaches correspond to a static set of rules that consequently fail to account for contextual integrity. Towards tackling this issue, researchers proposed context-aware permission managers [23]–[25]. The challenge here however, relates to the definition and modelling of context [18], while keeping the user interaction to a minimum. Therefore, to model context, simplified approaches are typically used to describe user state (e.g., time and location) and device status (e.g., screen state). In particular, visibility of the app [5], [24] has been reported as one of the most important contextual features guiding the users towards granting or denying a permission request.

Our research is more aligned with the work from [7], where the authors inquire through surveys whether users expect certain apps to require specific permissions, their degree of comfort in allowing such permission and for which purpose the app required the permission. From the responses, the authors concluded that the comfort of the user is highly related to their expectations, which in turn is influenced by the knowledge about the apps' requirements for the permissions. The strongest limitation of this work is that it was conducted with install-time permissions, in where users either accepted all permissions, or refuse to install the application. Therefore, it is inconclusive on how much expectation affects privacy decisions. Furthermore, it does not capture privacy's contextual subjectiveness, that is, it fails to capture how expectation varies with regard to the context of the phone [5], for example, the visibility of the requesting app. In fact, the time at which the request is accessed and the purpose of the access, the context, are crucial [12]. In this work we collected privacy decisions in-situ and at scale from 93 participants, including their respective expectation for each permission request and the surrounding contextual data. From this data we analyze the expectation of users regarding current app practices under the runtime permission model and evaluate its impact in personal privacy decisions.

We should note that our field trial was run with Android 9, the latest operating system at the time. Since our study, newer releases have improved the permission manager. Following research on the importance of app visibility in users' privacy decisions [5], Android 10 has introduced a new permission that is required for apps to access location when the app is in the background, thus enabling the user to allow access to the location permission only while using the app. In 2020, the permission manager in Android 11 implements: one-time permissions, which grant an app the permission a single time; permissions auto-reset, where granted permissions from apps are automatically set to the denied state when the app is unused for a few months; and automatically blocked permissions, for permissions that are always denied by the user for specific apps. The permission manager used in our campaigns, which is described in Section III-A more closely resembles the *modus operandi* of the permission manager from Android 11 as if the user always chooses one-time permissions, with the difference that in our permission manager user choice would be cached for 30 minutes and replayed should the app require the same permission during this time.

III. Methodology

In order to relate privacy decisions with the respective expectations, we perform data collection campaigns with a total of 93 volunteers from Portugal. Participants were recruited through word-of-mouth, university mailing lists and from oral presentations. This resulted in the participation of 60 (64.5%) students, 11 (11.8%) researchers and the remaining 19 (20.4%) with diverse backgrounds. Some 66 (71%) participants were between 18-24 years old, 25 (26.9%) between 25 and 39 and 2 (2%) over 40 years old. While most participants were students, the professional areas of occupation diverged: 53 (57%) participants were from informatics engineering or computer science, 12 (12.9%) from other engineer fields, 8 (8.6%) from exact sciences other than engineering and the remainder spread through other occupations, retired or did not answer the question. Therefore, the dataset is skewed towards young adults and slightly more than half with an IT background.

A campaign consisted of a period of at least 1 week, where participants used borrowed smartphones that came pre-installed with their personal apps, which we collected beforehand through our campaign signup app, along with Naive Permission Manager (NPM), our data collection tool and permission manager. NPM and the collected data are detailed in Section III-A.

Participants that showed interested in continuing the experiments were sometimes allowed to continue using the phone for longer. Participation was rewarded with a gift card with the requirement of using the campaign phone as the main smartphone for the duration of the campaign. While this requirement was explicitly announced, we gave the voucher to every participant.

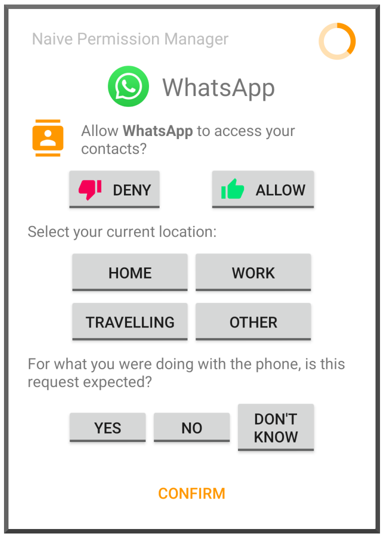
Naive Permission Manager
WhatsApp
Allow **WhatsApp** to access your contacts?
DENY
ALLOW
Select your current location:
HOME
WORK
TRAVELLING
OTHER
For what you were doing with the phone, is this request expected?
YES
NO
DON'T KNOW
CONFIRM

Fig. 1: An example of a translated (from Portuguese) permission prompt issued by Naive Permission Manager as a result of the app "WhatsApp" checking for the contacts permission.

A. Naive Permission Manager

To collect the data throughout the campaigns, the borrowed smartphone has our Naive Permission Manager (NPM) pre-installed alongside the participant's personal apps. NPM is both a permission manager and our data collection tool. Specifically, NPM intercepts *permission checks* performed by any app and prompts the user to either accept or deny the permission. At the time of the prompt, NPM further collects contextual data and additional information from the user as follows:

- Requesting Application: name, package name, version code, UID, flags and app category from the Play Store.
- Permission: the name and group of the permission and the user response.
- Phone state: geolocation, plug, dock, call, screen and keyguard states, network connection type, list of apps running in the foreground and in the background. An

application is in the foreground if it either has an activity in the foreground (visible to the user) or a service with a foreground notification. Apps running in the foreground and background have the same fields as the requesting application.

- User context: current time, semantic location, Bluetooth and WiFi devices in vicinity and whether the user is or is not in an event, as returned by their calendar. The semantic location was collected from user input, whose possibilities were "home", "work", "travelling" or "other" as illustrated in Figure 1.
- Expectation: the participant has to answer the question (translated from Portuguese) "For what you were doing with the phone, is this request expected?" with: yes, no or do not know. See Figure 1 for an illustration on how this data was asked to participants.

The permission dialog and context data are collected, stored locally and finally sent opportunistically to our project server.

1) Implementation Details: The Naive Permission Manager (NPM) comes preinstalled in campaign phones, acting as data collection tool and permission manager. Below we provide an implementation overview of NPM and refer the interested reader to the project website for further detail [26].

With runtime permissions, apps need to check if they have a permission before executing the API call that would collect the data. If an app does not have the permission, the app may or may not make a request [27], as it can check whether it has the permission without explicitly asking the user. This conditional execution led us to primarily intercept *permission checks*, escalating them to permission requests in the form of the prompts illustrated in Figure 1, while intercepting *permission requests* only to override the result with the participant input given to the prompt created by NPM at the respective permission check. This latter interception allow us to bypass the Android default permission manager. Note that while we collect the data at permission checks, we refer to this data as *permission requests*.

In order to not bias the data, NPM follows the Android system by only requesting *dangerous permissions* and managing permissions on a group level [27], that is, if an app requires the read calendar permission and the user grants it, the app will automatically be granted the write calendar permission on request. When a dangerous permission check call is made by an app, NPM prompts the user as illustrated in Figure 1 and collects the contextual data aforementioned. Similarly to the work in [25], we cache the answer for 30 minutes, thus returning the same answer for the given app and permission group for this duration, in order to avoid warning fatigue [28]. The permission icon and permission description are obtained directly from the Android operating system, so as to not bias the response. To avoid breaking functionality, NPM does not handle permission requests from system apps, letting the Android native permission manager handle those.

B. Dataset Sharing and Ethics

Due to the limitations in existing datasets, we make an anonymized version of our dataset available to interested researchers [29]. Please contact us for access or for potential collaborations. All shareable data is stripped of identifiable information. Our Naive Permission Manager is open-sourced and freely available [30].

This research was approved by the Ethics Committee, Department of Computer Science and Technology, University of Cambridge, and by the Ethics Commission of the Faculty of Sciences and Technology of the University of Porto.

IV. Dataset Characterization and Effect of User Expectation

From the 93 participants, we collected 2180302 permission requests at an average of 836.85 requests per day and per participant with a standard deviation (std) of 19.15, or 34.87 ($std = 0.8$) per hour. These numbers prove that an ask-on-every-time approach, the ideal privacy choice, is infeasible in practice. Note however, that this number varies with general phone usage, including the type of installed apps. Of the total requests, 65261 (2.99%) were answered by participants, corresponding to an average of 25 ($std = 0.42$) answers per day, per participant. Permissions not answered by the participant were either answered by the cache, timeouts or dismissed. The following subsections analyze these answers, where Section IV-A focuses on the grant result, that is, whether the user allowed or denied the permission, while Section IV-B delves into the effect of user expectation on privacy decisions.

A. Analysis of Grant Ratio and Privacy Violations

From the 65261 answered requests, participants granted 43263 (66.29%), while denying the remaining 21998 (33.71%); that is, users grant 2 out of every 3 permission requests. These results strongly contrast with the grant rate reported in [2], where participants granted 86% of requests. This disparity occurs due to the fact that the data in [2] was collected from Android's runtime permission prompts, which only occur when apps have their permissions denied and are running a foreground activity. However, after being allowed once, applications can access the resource any time even without the user being aware, until it is explicitly denied through the phone settings. Our permission dialog, on the other hand, prompts users on every *permission check*, unless the same permission has been answered in the last 30 minutes as previously explained, including from background apps, regardless of whether they previously had the permission granted.

Similarly to the results discussed in [2], our data reveals a strong variation of the grant ratio depending on the category of the requesting app and the requested permission. Specifically, the grant ratio varies mostly in the interval of [45, 75]% depending on the category and in the interval of [45, 85]% depending on the requested permissions, as displayed in GR information on the labels of the axis of Figure 3. CAMERA, STORAGE and CALENDAR permissions are granted over

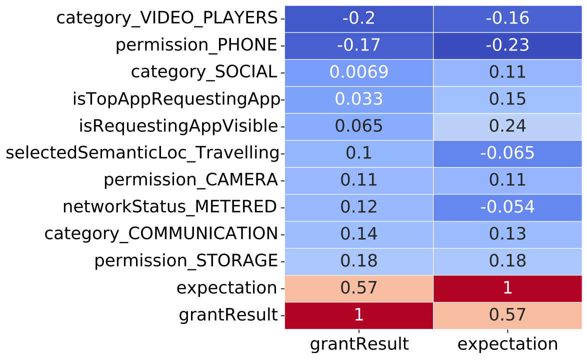

Fig. 2: Pearson correlation coefficient for the grant result and expectation with all other features, where categorical features are one-hot encoded, requests with UNKNOWN expectation value removed and coefficients in the interval of $]-0.1, 0.1[$ are omitted.

80% of the time, which might indicate that when apps request these permissions, there are contextual cues or a clear necessity that lead users to allow. We further discuss this aspect when analyzing the grant rate as a function of the expectation of the request.

It is possible to assess the number of privacy violations of the default Android Permission system. Specifically, we can measure the number of requests that would have been allowed by Android's permission manager but instead were denied by the user, as after being accepted once the permission is allowed everytime. We do so by counting the number of requests for each pair of app–permission that were denied after being accepted once. *From the 65261 user answered permissions, 9950 (15%) were denied by participants that would have otherwise been allowed by Android's permission manager.* Note that while this number seems rather small, this corresponds to privacy violations 15% of time. Additionally, due to the fact that the default Android permission manager would prompt a request until the user allows a permission for each specific app, achieving this violation rate with an Android system would require users answering an average of 129.5 permission prompts (median of 64). The privacy violation ratio is identical to the results from [24] (from their table III, 15.39% would be wrongly allowed). However, the number of prompts is significantly different, as we saw a median of 64 prompts per user, whereas [24] reported 12.34. This disparity is justified by the fact that their work came before the introduction of runtime permissions in Android, and therefore the set of dangerous permissions that the authors considered differs from our set, the default Android dangerous permissions.

From the 139440 requests where we have contextual data, 53938 (39%) were requests in where the requesting app was in the foreground and the remaining 85502 (61%) from the background, that is, approximately two thirds of the requests come from apps running in the background. Users allow 68% of requests coming from visible apps, while allowing 62% of requests from background apps. In contrast with the line of work in [5], [24], our values suggest that the visibility of the requesting app actually has limited importance in the privacy decision; the pair category-permission is more relevant. Figure 2 confirms this hypothesis, by showing the Pearson correlation coefficient for the grant result and all relevant features, where the visibility of the requesting app (isRequestingAppVisibility) sees a coefficient value close to 0, while some categories and permissions see a stronger correlation.

B. *Analysis of the Effect of User Expectation*

The strongest correlation with the grant result, that is, the privacy decision to either accept or deny the permission, is user expectation with a coefficient value of 0.57, as evidenced in Figure 2. This result can be further analyzed by looking at the distribution of the grant result for each expectation value, which can be EXPECTED, UNEXPECTED or UNKNOWN, where this latter value corresponds to when the user was unsure whether the request was expected or unexpected. Specifically, when users expect a request, they allow it 92% of the time, while allowing only 38% of the requests that are unexpected. When in doubt, the user accepts 2 out of 3 ($\approx$ 67%) requests, which is inline with the global grant ratio. These results indicate that developers should explain the rationale behind permission requests, a possibility implemented since Android 6.0 and iOS6, that has been shown to help with privacy decisions [11], yet it is still largely unused in practice [10].

From the 65261 user answered requests, 52% were EXPECTED, 46% were UNEXPECTED and the remaining 2% were UNKNOWN, that is, the participant was unsure whether the request was expected or unexpected. In other words, almost half of requests are unexpected. This result reveals a strong misalignment between app practices and the expectation of users, and therefore calls for more transparency from app developers, as an informed user is a comfortable user [7], and endorses the use of the minimum required permissions for the functionality of the app.

Similarly to the grant result, user expectation varies for each category and each permission. Figure 3 presents the average expectation for each pair of category-permission and the grant and expected ratios, where the latter is the percentage of EXPECTED requests, for each category and permission. It is clear from the plot that some pairs of category-permission are often expected, such COMMUNICATION-STORAGE, while others are often unexpected, such as GAME-PHONE. In fact, the PHONE permission sees the lowest expected ratio from all permissions at only 22.3%, closely followed by the CALL_LOG permission with 23.6% of expected requests. Our reasoning for these low values lies on their lack of understanding surrounding these two permission groups. The CALL_LOG permission group was created in Android 9 by moving some of the PHONE permission to the former group. At the time of the study it is possible that some users did not have Android 9 in the personal phones and were therefore first exposed to this permission group during the campaign.

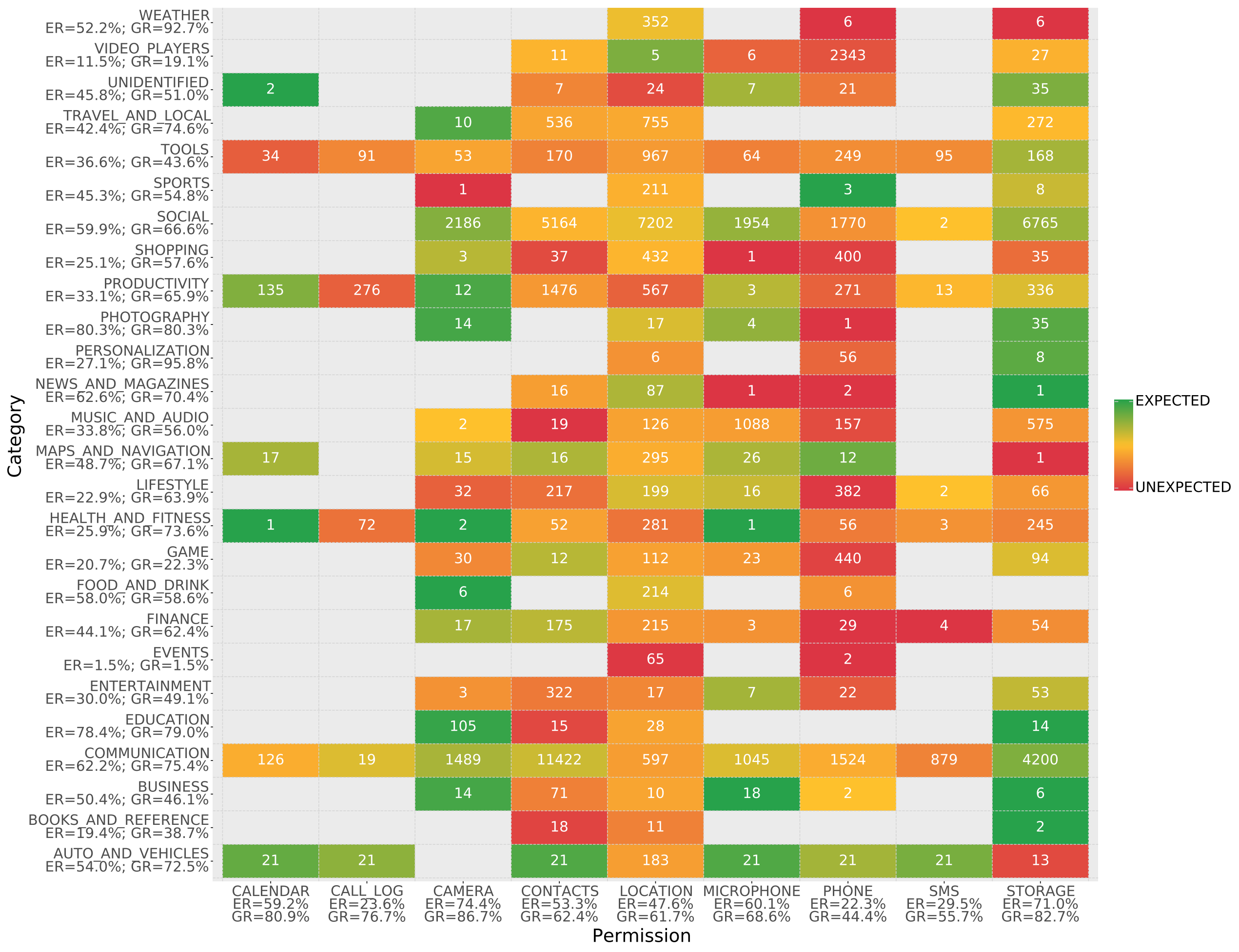

Fig. 3: Average expectation for each pair of category-permission, with requests with UNKNOWN expectation removed. The number in each cell is the number of requests for the respective pair category-permission group and ER and GR are respectively the expected ratio (percentage of requests that were expected) and grant ratio for the respective category/permission. Categories and permissions with less than 10 requests were removed.

Furthermore, the PHONE permission allows not only checks on the phone state, but also to make and manage calls and even access unique identifiers[1], functionalities that might not be evident to the user. In fact, previous work [28] showed that less than 5% of 85 respondents correctly identified the functionality of READ_PHONE_STATE, a permission within the PHONE group. Almost as low in the expectation ratio as the PHONE and CALL_LOG, comes the SMS permission with less than 30% of requests expected, as illustrated in Figure 3. Contrary to the other two permissions, the functionalities allowed by the SMS group are arguably clearer [28]. A possible reasoning for this low expectation lies in the number of SMS requests that originate from the background. Only 28% of these requests (c.f. in Figure 4) originate from apps that are visible to the user, making the user unsure on the need for these requests. It is possible that the functionality provided by this permission group is not worth for the user. Confirming this would require a survey. However, the sensitivity of the SMS and CALL_LOG groups has led Google to restrict their usage to the default SMS/Phone/Assistant handler or as core app features [32].

[1]Access to unique identifiers is now restricted since Android 10 [31].

In contrast with the limited influence of the visibility of the requesting app in the grant result as discussed in Section IV-A, our data reveals that the expectation is influenced by it, as showcased by a correlation coefficient value of 0.24 in Figure 2. Particularly, approximately 60% of requests originating from a foreground app are expected, whereas only $\approx 34\%$ are expected from background apps, that is, 2 out of every 3 requests originating from background apps are unexpected. This ratio greatly varies between the different categories and permissions, were the values for the latter are illustrated in Figure 4. From this plot we observe that for any permission, it is more likely to be expected when requested from a foreground app than from a background app. Additionally, most requests for both SMS and CALL_LOG permissions come from the background, where in these situations, less than 20% are expected, while STORAGE, MICROPHONE and CAMERA were requested from the foreground over 80% of times, where the expectation ratio in these cases was over

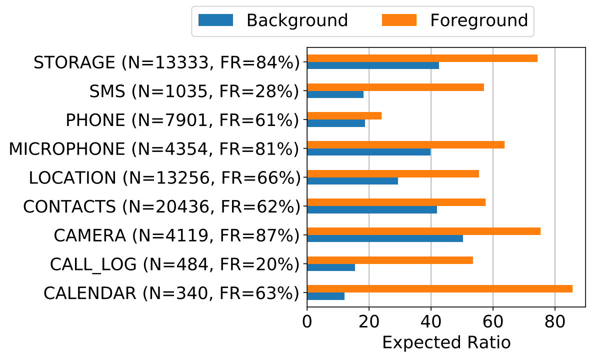

Fig. 4: Expected ratio per permission group and visibility of the requesting app. The "N" is the number of requests per group and "FR" the foreground ratio, that is, the percentage of requests that came from apps that were visible to the user at the time of the request.

60%.

Privacy expectations can be highly subjective due to the imperfect mental model (knowledge) that each individual has about the functionalities of apps [7]. An open question however, is how much influence does expectation have on privacy decisions. Previous results showed than in general expected requests are allowed and unexpected requests are mostly denied. However, this can differ for each participant. Figure 5 presents the average grant result per expectation value, for each user. In this plot, users are represented by their ID on the x axis, while the respective average grant result is presented as a colored bar for each expectation value on the y axis. The color ranges from dark red, if the user denies all requests, to dark green if the user allows all. For example, the user with ID 22 mostly rejects requests independently of whether they are expected or not, while the user with ID 61 allows all expected and unknown requests, while denying all unexpected ones. From this plot we observe that the importance of expectation greatly varies with each individual. There are users whose privacy decisions are uncorrelated with their expectations. In the plot, these are the users with similar color bars for any expectation values, and we see examples of users that allow all (all green), deny all (all red) or allow or deny selectively (all orange/yellow). Then, there are people that deny most or all unexpected requests but allow most or all expected, as can be seen by the green bar in expected requests and red/orange in unexpected. These are individuals whose privacy decisions closely follow their expectations and correspond to the majority. Finally, there are participants in between the previous two extremes, which take into consideration the expectation as well as other variables, such as the visibility, the category or requested permission. In summary, while the importance of expectation in privacy decisions varies for each user, the majority acts in accordance with their expectations, as highlighted from the strong correlation in Figure 2 and now confirmed in Figure 5. We pose the possibility of modeling the expectation towards automating privacy decisions with fewer privacy violations. We leave such an endeavor for future work.

V. Limitations And Future Work

The data was collected in a set of campaigns spawning from July 2020 up to May 2021 in Portugal. This period included periods of mandatory confinement and recommended remote work, thus limiting the data collected at each (semantic) location. In fact, both the collected context and app usage might differ from *normal* conditions. We should note that we balanced the dataset for each location and verified that the insights obtained in this work hold. We leave as future work any analysis of the impact of the COVID19 in the data.

Borrowing a campaign phone has the disadvantage of having to configure participant applications on the phone. To ease transition and favor using the campaign phone, we installed the participant's personal apps on the campaign phone before lending. The advantage, however, is that any person can participate in our experiments. Using personal phones would be possible, but CM-NPM requires administrator permissions (rooted Android device), which would reduce the experiment population and bias the dataset towards more tech-savvy participants. Nevertheless, participants were still required to configure their accounts in each app. Due to the short duration of the campaigns, some participants might have not configured all apps, potentially limiting the amount of data collected.

To enhance the overall quality of the dataset and ecological validity of the findings we could have collected app usage from the personal phone, although doing so would require rooted personal phones due to Android's restrictions, and implement opportunistic surveys to further analyze the reasoning behind the expectation and respective privacy choices. We leave these remarks as learnt lessons for future works.

VI. Conclusion

Runtime permission managers allow permission requests to be contextualized by the need to access functionality at the time of the prompt. However, after being accepted once, subsequent requests are automatically granted without user awareness and therefore can violate user expectation. In this work we performed a field study with 93 participants to collect permission decisions, the surrounding context and the expectation of the user regarding each request at runtime and in-situ. This data reveals a strong misalignment between apps practices and user expectations. In fact, almost half of requests are unexpected. Additionally, there is a strong correlation between privacy decisions and the expectation of the user. In particular, 9 out of 10 expected requests are allowed, a ratio that highlights the importance of explaining app requirements to the user. However, both the expectation and the importance of the expectation in the decision are highly personal. Finally, our dataset reveals that the default Android system would have violated user privacy in 15% of requests. The high correlation observed between expectation and grant rate calls for methods to model user expectation in order to predict grant decisions with minimal user input. Furthermore, our results should spark further research on improving privacy in mobile devices by taking into consideration the expectation of the user in the design of applications.

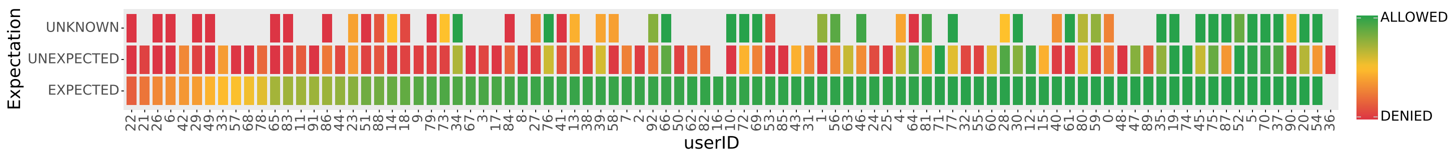

Fig. 5: Average grant result per expectation value and per user. Each user is represented by an id on the x label and the average grant result is represented as a color ranging from dark green, if the user allows all requests, to dark red, if the user denies all requests, for each of the three expectation values in the y axis.

References

[1] L. Cranor, T. Rabin, V. Shmatikov, S. Vadhan, and D. Weitzner, "*Towards a Privacy Research Roadmap for the Computing Community*: A white paper prepared for the computing community consortium committee of the computing research association." 2015.

[2] B. Bonné, S. T. Peddinti, I. Bilogrevic, and N. Taft, "Exploring decision making with android's runtime permission dialogs using in-context surveys," in *Thirteenth Symposium on Usable Privacy and Security (SOUPS 2017)*. Santa Clara, CA: USENIX Association, 2017, pp. 195–210.

[3] P. Andriotis, G. Stringhini, and M. A. Sasse, "Studying users' adaptation to Android's run-time fine-grained access control system," *Journal of Information Security and Applications*, vol. 40, pp. 31–43, 2018.

[4] H. Almuhimedi, F. Schaub, N. Sadeh, I. Adjerid, A. Acquisti, J. Gluck, L. F. Cranor, and Y. Agarwal, "Your location has been shared 5,398 times!: A field study on mobile app privacy nudging," in *Proceedings of the 33rd annual ACM conference on human factors in computing systems*. ACM, 2015, pp. 787–796.

[5] P. Wijesekera, A. Baokar, A. Hosseini, S. Egelman, D. Wagner, and K. Beznosov, "Android permissions remystified: A field study on contextual integrity." in *USENIX Security*, vol. 15, 2015.

[6] H. Nissenbaum, "Privacy as contextual integrity," *Wash. L. Rev.*, vol. 79, p. 119, 2004.

[7] J. Lin, S. Amini, J. I. Hong, N. Sadeh, J. Lindqvist, and J. Zhang, "Expectation and purpose: understanding users' mental models of mobile app privacy through crowdsourcing," in *Proceedings of the 2012 ACM conference on ubiquitous computing*, 2012, pp. 501–510.

[8] A. Barth, A. Datta, J. C. Mitchell, and H. Nissenbaum, "Privacy and contextual integrity: Framework and applications," in *2006 IEEE symposium on security and privacy (S&P'06)*. IEEE, 2006, pp. 15–pp.

[9] A. Acquisti, L. Brandimarte, and G. Loewenstein, "Privacy and human behavior in the age of information," *Science*, vol. 347, no. 6221, pp. 509–515, 2015.

[10] X. Liu, Y. Leng, W. Yang, W. Wang, C. Zhai, and T. Xie, "A large-scale empirical study on android runtime-permission rationale messages," in *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 2018, pp. 137–146.

[11] J. Tan, K. Nguyen, M. Theodorides, H. Negrón-Arroyo, C. Thompson, S. Egelman, and D. Wagner, "The effect of developer-specified explanations for permission requests on smartphone user behavior," *Proceedings of the 32nd annual ACM conference on Human factors in computing systems - CHI '14*, pp. 91–100, 2014.

[12] D. Votipka, S. M. Rabin, K. Micinski, T. Gilray, M. L. Mazurek, and J. S. Foster, "User comfort with android background resource accesses in different contexts," in *Fourteenth Symposium on Usable Privacy and Security (SOUPS 2018)*, 2018, pp. 235–250.

[13] I. Shklovski, S. D. Mainwaring, H. H. Skúladóttir, and H. Borgthorsson, "Leakiness and creepiness in app space: Perceptions of privacy and mobile app use," in *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*. ACM, 2014, pp. 2347–2356.

[14] B. Shen, L. Wei, C. Xiang, Y. Wu, M. Shen, Y. Zhou, and X. Jin, "Can systems explain permissions better? understanding users' misperceptions under smartphone runtime permission model," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021.

[15] M. Harbach, M. Hettig, S. Weber, and M. Smith, "Using personal examples to improve risk communication for security & privacy decisions," in *Proceedings of the SIGCHI conference on human factors in computing systems*. ACM, 2014, pp. 2647–2656.

[16] F. Shih, I. Liccardi, and D. Weitzner, "Privacy tipping points in smartphones privacy preferences," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, 2015, pp. 807–816.

[17] Y. Feng, Y. Yao, and N. Sadeh, "A design space for privacy choices: Towards meaningful privacy control in the internet of things," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, ser. CHI '21. New York, NY, USA: Association for Computing Machinery, 2021.

[18] F. Schaub, R. Balebako, A. L. Durity, and L. F. Cranor, "A design space for effective privacy notices," in *Eleventh Symposium On Usable Privacy and Security (SOUPS 2015)*. USENIX Association, 2015, pp. 1–17.

[19] J. Gluck, F. Schaub, A. Friedman, H. Habib, N. Sadeh, L. F. Cranor, and Y. Agarwal, "How short is too short? implications of length and framing on the effectiveness of privacy notices," in *Twelfth Symposium on Usable Privacy and Security (SOUPS 2016)*, 2016, pp. 321–340.

[20] B. Liu, M. S. Andersen, F. Schaub, H. Almuhimedi, S. Zhang, N. Sadeh, A. Acquisti, and Y. Agarwal, "Follow my recommendations: A personalized privacy assistant for mobile app permissions," in *Symposium on Usable Privacy and Security*, 2016.

[21] Y. Elbitar, M. Schilling, T. T. Nguyen, M. Backes, and S. Bugiel, "Explanation beats context: The effect of timing & rationales on users' runtime permission decisions," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 785–802.

[22] B. Liu, J. Lin, and N. Sadeh, "Reconciling mobile app privacy and usability on smartphones: could user privacy profiles help?" in *Proceedings of the 23rd international conference on World wide web - WWW '14*, 2014, pp. 201–212.

[23] P. Wijesekera, J. Reardon, I. Reyes, L. Tsai, J.-W. Chen, N. Good, D. Wagner, K. Beznosov, and S. Egelman, "Contextualizing privacy decisions for better prediction (and protection)," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ser. CHI '18. New York, NY, USA: ACM, 2018, pp. 268:1–268:13.

[24] P. Wijesekera, A. Baokar, L. Tsai, J. Reardon, S. Egelman, D. Wagner, and K. Beznosov, "The Feasibility of Dynamically Granted Permissions: Aligning Mobile Privacy with User Preferences," in *Proceedings - IEEE Symposium on Security and Privacy*, 2017, pp. 1077–1093.

[25] K. Olejnik, I. Dacosta, J. S. Machado, K. Huguenin, M. E. Khan, and J.-P. Hubaux, "Smarper: Context-aware and automatic runtime-permissions for mobile devices," in *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2017, pp. 1058–1076.

[26] R. Mendes, "The Project COP-MODE," https://cop-mode.dei.uc.pt/, 2021, accessed: 2022-01-14.

[27] G. Developers, "Request app permissions," https://developer.android.com/training/permissions/requesting, accessed: 2021-08-29.

[28] A. Felt, E. Ha, S. Egelman, and A. Haney, "Android permissions: User attention, comprehension, and behavior," *Proc. of SOUPS*, pp. 1–14, 2012.

[29] R. Mendes, "COP-MODE Dataset Guide," https://cop-mode.dei.uc.pt/dataset, 2021, accessed: 2022-01-20.

[30] ——, "COP-MODE Naive Permission Manager," https://cop-mode.dei.uc.pt/cm-npm, 2021, accessed: 2021-08-29.

[31] Android Developers, "Privacy changes in android 10," https://developer.android.com/about/versions/10/privacy/changes#non-resettable-device-ids, 2019, accessed: 2022-01-07.

[32] Google, "Use of sms or call log permission groups," https://support.google.com/googleplay/android-developer/answer/10208820, 2018, accessed: 2022-01-07.