# Investigating Users' Reaction to Fine-Grained Data Requests: A Market Experiment

Nicole Eling
TU Darmstadt
eling@is.tu-darmstadt.de

Siegfried Rasthofer
TU Darmstadt
siegfried.rasthofer@cased.de

Max Kolhagen
TU Darmstadt
max.kolhagen@stud.tu-darmstadt.de

Eric Bodden
Fraunhofer SIT & TU Darmstadt
eric.bodden@sit.fraunhofer.de

Peter Buxmann
TU Darmstadt
buxmann@is.tu-darmstadt.de

## Abstract

*The market for smartphone applications is steadily growing. Unfortunately, along with this growth, the number of malicious applications is increasing as well. To identify this malware, various automatic code-analysis tools have been developed. These tools are able to assess the risk associated with a specific app. However, informing users about these findings is often difficult. Currently, on Android, users decide about applications based on coarse-grained permission dialogs during installation. As these dialogs are quite abstract, many users do not read or understand them. Thus, to make the more detailed findings from security research accessible, new mechanisms for privacy communication need to be assessed. In our market experiment, we investigate how fine-grained data requests during runtime affect users' information disclosure. We find that many users reverse their decision when prompted with a fine-grained request. Additionally, an effect of security awareness and level of detail on disclosure was found.*

## 1. Introduction

The market for smartphone applications is steadily growing with a total number of more than 1.5 Mio apps in Google Play [1]. These applications provide users with various features facilitating their daily lives, ranging from banking applications over weather apps to fun applications. While these applications provide various benefits, there are also serious risks associated with their usage. When an app is installed on a smartphone, depending on the operating system the application obtains various permissions allowing it to access all kinds of user information. Further, there is a vast amount of malware utilizing security gaps to harm users [2, 3].

Previous research has shown that users have often difficulties assessing the impact of information disclosure. Especially in the mobile app context, many users do not understand the permissions and potential risks associated with app installation [4, 5]. Due to this lack of knowledge, users might underestimate the risks, and therefore grant permissions they otherwise would not have given. In addition, research has found that users discount potential future risks and put higher emphasis on the short-term benefits of information disclosure [6, 7]. As privacy and security are greatly threatened, we want to explore if this assessment changes, when information is made more explicit. Especially for users with little security awareness and knowledge a more detailed, easier to understand, imminent information request might reduce their likelihood to reveal their information. This work thus explores the following research questions using a self-developed mobile application:

1) How does the precision of an information request influence users' disclosure of personal information?
2) Is this effect different for users with different security backgrounds?

These research questions are investigated using data obtained through a smartphone app offered in Google Play. By doing so, we meet the call for measuring real behavior instead of stated willingness to disclose [8]. This is important as users' intentions often differ from user behavior in the context of privacy [9, 10].

Our paper is structured as follows. First, we give a short overview of relevant literature. Then, we derive our hypotheses which are later tested within our study. When presenting our methodological approach, we discuss the advantages and disadvantages of using a real application and introduce our research design. In chapter five, we present the results. Finally, we discuss our findings and conclude with limitations and avenues for future research.

## 2. Related work

In this section, we describe related work in the context of Android malware analysis that addresses threats and mitigation mechanisms. Furthermore, we present findings in the field of information privacy focusing on user risk perceptions and privacy behavior.

There are different kinds of malicious applications in circulation [2], such as apps that send SMS messages to premium-rate numbers or steal banking data or location information. While the engineering effort for Android malware was very low in the beginning, recently, it is getting more and more sophisticated [3]. The Android OS itself provides different security protection mechanisms such as the internal permission model which shows the user what resources are accessed by the application, for instance, contact data or text messages. However, research has shown that users often do not understand the permissions [4, 5]. One reason is that the permission system is too coarse-grained and provides too little security information about *what* data are sent *where* and *why*. To address this issue, CRePE by Conti et al. [11], Apex by Nauman et al. [12] or DroidForce by Rasthofer et al. [13] have proposed a fine-grained permission model to replace Android's permission model. In these approaches, a user or a trusted entity is able to define fine-grained policies such as *Application X is not allowed to send my girlfriend's contact data to server S before 6pm*, which get enforced during runtime. While these approaches are very promising due to their fine-grained nature, it is still an open research question if non-security experts are able to define such precise policies in a correct way or whether it overtaxes the user.

The techniques underlying these approaches are static and dynamic code analysis approaches, or a combination of both. In addition to applying them to the permission problem, they are also essential in automatically detecting malicious applications [3]. Static analysis is a code-analysis approach that can analyze all paths in the application's code (bytecode or source code) without executing it, while dynamic analysis executes the application in order to get concrete runtime information about the execution. Both have their pros and cons, but are heavily used in a security context [3]. For instance, if an analyst needs to know whether an application leaks some sensitive data to a specific server, one can use static or dynamic code analysis approaches to answer this question in a fully-automatic way. In the context of Android, FlowDroid by Arzt et al. [14] is a static data-flow tracking tool that scans an Android application for identifying data leaks before it gets installed on the device. A dynamic analysis approach for identifying data leaks is TaindDroid [15], which requires the execution of the application and informs the user about a potential data leak during runtime. Another important field where static and dynamic code analysis approaches are heavily used is the detection of vulnerabilities in applications. CHEX [16] and Mallodroid [17] are static analysis tools that try to identify specific vulnerabilities in Android applications. In comparison, Sounthiraraj et al. [18] combines both static- and dynamic analysis approaches (hybrid code analysis) in order to identify SSL vulnerabilities.

When considering security and privacy, to identify risks it is very important not only to consider the approaches presented above but also to be aware of users' risk perceptions and the resulting user behavior [19]. There is various research on risk in general investigating risk's role in an individual's decision making [20]. Transferring this general perspective to privacy, research has found that users often weigh benefits against perceived risks in a so called privacy calculus when deciding about information disclosure [21]. When benefits exceed risks, the permission request is accepted. Research investigating risk perceptions in this context have among others analyzed how these perceptions are formed [22], and how risk perceptions and consent dialogs influence users' willingness to accept information requests [23-25]. For instance, research in the context of mobile apps has found that users undervalue the probability of risk [22].

The research community behind code-analysis techniques has engineered very sophisticated techniques to identify data leaks or similar threats, whereas the privacy community has very good experience in users' risk perceptions. This work seeks to take a first step to close the gap between these two communities and to measure how sensitive data leaks can be reported in a way that the user understands the privacy and security impact.

## 3. Hypotheses

Research on privacy policies and app-permission dialogs has shown that users often do not read or understand them [4, 5]. This lowers the users' ability to make an educated judgement about potential privacy risks. Further, associated risks are often not transparent and lie very far in the future. Due to this, they are discounted and valued less than the short term benefits of disclosure [6, 10]. This behavior is further encouraged by the permission dialogs used in Android operating systems. Users have to decide about permissions before installation. As it is very hard to assess the relevance of the permissions and potential benefits and risks of disclosure beforehand, this procedure makes it very hard to make a decision corresponding with users' privacy attitude.

The main research hypothesis of this paper is that risks would be more imminent if information access was made more explicit by presenting fine-grained permission requests. Fine-grained permissions can have a much higher degree of accuracy than coarse-grained permissions. For instance, Android's permission model provides coarse-grained permissions such as "Internet Access", where a fine-grained permission would be "Access to www.malicious.com".

Further, we pose that it would be easier for individuals to weigh benefits and risks when the concrete benefits associated with disclosure were more transparent. This can be achieved by displaying the data request at the time of data access. This way the abstract and unnoticed data transfer would become more obvious and contextual. Disclosure would not be an abstract risk or some unknown process anymore, but would instead happen directly in front of the user's eyes. When users become aware of the data that is transferred, they may therefore be less willing to share the information and reject the data request. We therefore hypothesize that:

*H1: A fine-grained permission request during runtime is less likely to be accepted than a generic permission request before installation.*

While users might understand fine-grained permission requests more easily, the risk might still be abstract to some extent. Displaying concrete user information instead of the abstract name of the data type might increase the users' attention and increase their risk perception resulting in a decreased willingness to disclose their data:

*H2: A data request containing concrete user information reduces the user's likelihood to accept it.*

Previous research has shown that knowledge and experience can have a great influence on privacy concern and on risk perceptions [8, 26]. We therefore hypothesize that users that are more security aware and who have dedicated measures in place to protect their smartphones are also more likely to reject information requests.

*H3: Security aware users are less likely to accept data requests.*

Further, as security-aware users are more likely to understand permission requests and their consequences during installation, we expect that security-aware users are influenced less by the level of detail of the request than security unaware users:

*H4: Security awareness moderates the effect of the level of detail of the information requests on information disclosure.*

## 4. Methodology

To test the hypotheses, we conducted a market experiment. In the context of privacy, intention to disclose as often measured in surveys and actual disclosure often do not match [9, 27]. Therefore, privacy researchers increasingly demand new methods that focus on real user behavior [8]. We attempted to meet this request by developing a smartphone app. We aimed at creating a situation as close to real life as possible. This is why instead of conducting a laboratory experiment, we uploaded our self-developed app to Google Play and measured user behavior within the app. This had the great advantage that users treated and assessed the app in the same way as any app downloaded from the app store. However, this procedure has the disadvantage that only little additional information about the users can be obtained without raising ethical concerns. Further, we did not have any influence on the selection of participants. But even considering these drawbacks, we believe that this realistic setting compensates for these disadvantages, and provides us with a unique opportunity to study information disclosure in the mobile app context.

The app was designed based on three criteria: Previous research has found that users perceive permissions differently when they might potentially be needed for the functionality of the app [28]. As we were not able to control for these perceptions through a survey, one criterion was to ensure that the functionality of the app was clearly independent of the permissions and information items requested. Another criterion was the potential relevance or interest to a wide variety of people. We did not want to target one specific group of users. An additional criterion was to keep the app simple and easy to use.

Trying to meet these criteria, we decided to implement a fun app, which belongs to the entertainment category. According to a recent study [29], entertainment apps resemble the second-most downloaded app category. These apps have the potential to spread very fast and to attract all kinds of users. In addition, there is not much functionality to which the need for different information items could be attributed. In contrast, if we had developed an app similar to Spotify, Facebook or a gaming application (most downloaded applications) which are used by a wide variety of users, there would have been a huge amount of potential influences on the users when assessing the necessity of different information items. While a fun app is not as serious as an app increasing usefulness, we think that especially these apps are critical in respect to permissions.

The functionality of our app was quite simple. It played all kinds of funny noises. Thus, it was a pure fun-app and its usefulness was very low. However, these kinds of apps have the potential to attract an extensive user base very fast. Apps offering similar functionality achieve up to five million users in the Google Play Store. As these kinds of apps can contain harmful malware [30, 31] user behavior when handling these apps is of special interest. While we acknowledge that the functionality is not very educated, we still believe that due to these reasons it has been a good choice to select this functionality.

When downloading the app, the user is first displayed a short tutorial, illustrating the functionality of the app. Then, the user is presented with the main screen displaying buttons with pictures. When a button is pressed, the app plays the corresponding sound. The main screen of our app is illustrated in Figure 1.

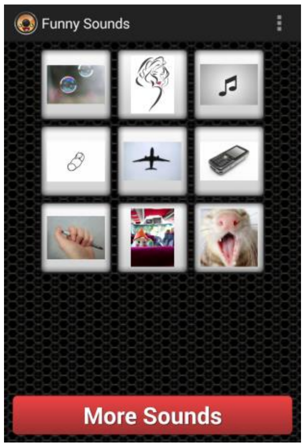
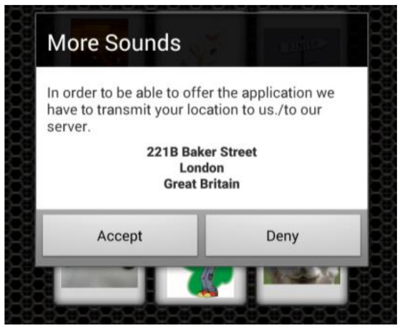


**Figure 1. Screenshot of the main-screen of the app**

Critically, to obtain new sounds, a user can push on a button called “More Sounds”. After pressing the button, the user is prompted to disclose some kind of personal information to get a new sound. This part represents the core of the experiment. Users were asked to trade their personal data for a very low benefit, namely a single new sound. While for ethical reasons user data was not transmitted to our server, users were made to believe that the information was in fact transferred. Information requests were assigned randomly to the users. In case users rejected this request, they were redirected to the main screen and did not receive the sound. If they accepted the request, it looked as if their information was transferred to our server, and a new sound-button appeared. For more sounds, users had to click the button again and to decide about a new information request. Figure 2 illustrates an exemplary request.


More Sounds

In order to be able to offer the application we have to transmit your location to us./to our server.

221B Baker Street
London
Great Britain

Accept | Deny

**Figure 2. Exemplary information request**

Each statement presented to the user consisted of four parts: The first part contained the reason why the provider claimed to collect the information. The second part displayed the information type, e.g. location data. The third part of the statement contained information about the target of the transmitted data, either the app provider or unknown third parties. Finally, we varied the granularity of the information request, i.e., an abstract statement stating that personal data is transmitted or a statement with concrete personal information displayed to the user.

In respect to the reasons for data request, we varied among financing, app improvement, app provision, and personalization. Regarding the data sink, we varied between data transfer to the provider and data transfer to third parties. We differentiated among five different data types: 1) location represents a data type often discussed in literature [32, 33]. Smartphones enable providers to collect real-time location information about their users. This information can be used to provide additional value by customizing the service to the user’s location [34]. However, there is also a great potential for misuse [35]. As a result, the privacy threats associated with the usage of location-based services are one of the biggest factors inhibiting their adoption [36]. 2) The phone number was used to model identifiable information. A phone number is unique and can be matched to a specific person. Therefore, revealing one’s phone number can be compared to revealing one’s identity. 3) We selected photos as third data type because they represent very sensitive information. In contrast to photos uploaded on social-networking sites, photos on a smartphone have not been pre-selected by the user and thus, might be especially sensitive. 4) Data about contacts was selected as it is especially interesting to see how users would decide about data of their friends, a research field hardly investigated so far [37]. 5) Finally, we made the users believe that we transmit their personal SMS. This is a very common type of malware where incoming SMS are intercepted and silently transmitted to the attacker, i.e., the user does not recognize the theft [2]. However, in our case, we displayed a message to the user saying that SMS messages were transmitted to our server.

The main focus of our experiment lay on the effect of the level of detail of the request on the users' acceptance (H2). We hypothesized that a statement involving the real information to be transferred would increase users' likelihood to reject the request. In the abstract message, we used the generic name of the data type, i.e., the statement would say that it requested the user's phone number. For more explicit messages, we used the information available on the smartphone. Thus, the user would not be presented with the generic name of the data type but with his true phone number. For instance the statement would say: "In order to be able to offer the application we have to transmit your phone number: 0123456 to us."

To do so, during installation of the app we had to obtain all permissions for accessing the information. Thus, users had actually agreed to everything beforehand and should be willing to accept every request. However, previous research has shown that users often do not pay close attention to permissions or do not understand them [38, 39]. Therefore, we expected that users would change their mind when prompted with more explicit statements as modelled by our app (see H1).

The different types of information were made more explicit as follows: To make location information more explicit and personal, in one variation we displayed the user's current address and location on a map. For the phone number, we displayed the user's real phone number. To illustrate that we were really going to transfer photos, contacts, and SMS, we randomly chose three photos/contacts/SMS saved on the phone and displayed them to the user.

The four different components were randomly combined to create the requests presented to the users. However, one restriction was that each user always received the same reason, as it would not have been logical if reasoning had varied across requests. Further, a user was never asked for the exact same request twice. Table 1 shows the different sentence fragments used to form the data requests. Depending on the final manipulation, only the sentence as shown in the table was displayed or we additionally displayed the personal information of the users directly extracted from their phone.

**Table 1. Sentence fragments**

| **Reasoning** | | **Data type** | | **Internal /external** |
|---|---|---|---|---|
| In order to finance our application… | | …we have to transmit your location… | | …to us. |
| In order to be able to offer the application… | | …we have to transmit your phone number… | | …to our partners. |
| In order to improve our app for you further… | + | …we have to transmit your photos… | + | |
| In order to be able to offer you personalized sounds… | | …we have to transmit your contacts… | | |
| | | …we have to transmit your text messages… | | |

When implementing the app, we took great care that the app would have an appealing design and that it made a serious impression. Our app was offered in three languages, English, Spanish, and German, so that it could be offered in many countries. This way, we also increased the likelihood that the request for information was understood by the participants.

We took great care to meet ethical standards. In the app store, we provided a link to a privacy policy, informing users about the data accessed as well as the use of the data. Users who did not want to participate could refrain from installing the app. Users were able to uninstall the application at any time. As we did not want to infringe users' privacy, we refrained from collecting any personal user information. While the app pretended that user data was transferred, no personal user data ever left the users' phones.

Before we uploaded the app, we conducted a pre-test with privacy and security researchers. In this pre-test, we asked participants to install and test the app. Feedback was integrated and the app was uploaded to the Play store at the end of May 2014. New apps are ranked very low by Google Play. Therefore, we asked our colleagues and friends to download and promote the app. As some of them knew about the experiment and we were not able to identify unique users, all users downloading the app during this initial phase were excluded from the consecutive analysis. Overall, the app remained in Google Play for a year. In this time, in total, 338 users downloaded the app. As described before, of these, all 75 users downloading the app in the initial phase (before July 2014) were excluded. Of the 263 remaining users, 73.38 % tried to add at least one new sound resulting in 193 valid first decisions. When users added several sounds, they had to make more than one decision, which is why our complete dataset comprises 596 user decisions.

## 5. Results

### 5.1. Measures

When a user started the app for the first time, we obtained data about his smartphone settings. However, to avoid invading the users' privacy, we refrained from collecting any information that could identify the user. The thus obtained data was used, to derive a proxy for determining users' security awareness. We extracted five different kinds of settings from the users' smartphone: Android OS version, device encryption mode, screen lock, installed apps that are security-related, and security flags. We assigned a value between zero and one to each of these indicators and then calculated the user's security awareness based on different weights assigned to those indicators.

Information about the Android OS is an important indicator since if the latest version is installed the user is protected against any vulnerability that has been discovered in previous versions. However, older smartphone generations might not be updated because they are no longer actively supported. Therefore, we decided to rate the security level three-folded (newer version available for user's smartphone: 0; most recent Android version for user's smartphone: .5; most recent Android version available: 1).

Encrypting the local storage of a device is a good indicator for a higher security awareness level. Since Android 3.0, users can manually activate device encryption. However, all versions higher than Android 5.0 support encryption by default. Versions lower than 3.0 do not support disk encryption at all. We applied three security-levels (encryption not activated: 0; encryption turned on by default: .5; encryption manually activated: 1).

The next indicator was the user's screen lock. This can be set to none, slide, face unlock, pattern, pin, or password. These modes vary in respect to their level of security [40, 41]. The indicator is rated according to this security level (none: 0; slide: 0; pattern: .33; face unlock: .66; pin: 1; password: 1).

Another important factor for measuring a user's security awareness is the installed applications. We extracted all information about the installed applications from the phone and matched them with security-related applications. (A list of applications can be requested from the authors.) The list contained at least all top ten applications from every security category and can be considered as representative. Security levels are set depending on the number of security-relevant apps installed (none: 0; one: .25; two: .5; three: .75; four and more: 1)

The last factor for security awareness was the activation of Android's internal security flags: "unknown sources" which allows the installation of apps from any source instead of just the official Google Play store (activated: 0; deactivated: 1) and "verify apps" which causes Android to analyze installed applications for well-known malware (deactivated: 0; activated: 1).

Overall, we rated the Android OS version, device encryption, screen lock and installed apps equally with 22.5 %. Security flags were rated only with 5 % each, as they are less reliable because also security-unrelated reasons for enabling/disabling them exist. For an exemplary user with the following settings, security awareness would be determined as follows: Version Android 4 where Android 5 is the latest version (.5 * 22.5 %) + device encryption is on per default (.5 * 22.5 %) + pin lock (1 * 22.5 %) + three security-related applications installed (.75 * 22.5%) + unknown sources flag enabled (0 * 5 %), + verify app flag enabled (1 * 5 %) = .67.

In addition to security awareness, every time a user was shown one of the data requests (see Figure 2), we noted and measured what data request was displayed, how long the user looked at the screen and if the user accepted or rejected the request.

### 5.2. Analysis

The user information requested in our experimental setting was information for which we had already obtained sufficient Android permissions during installation. Thus, we actually would not have had to ask the users for their consent again, and could have just transferred the data to our server. Therefore, any user rejecting the permission request reversed his or her original decision, providing evidence for H1.

Table 2 displays the total number of rejects and accepts in our sample. As mentioned earlier, each time users wanted to add a new sound, they had to give their consent for one specific data request. Only if they accepted this request, did they receive a new sound. The next time they wanted to add a new sound again, a different data request was displayed. For instance, when trying to add a new sound the first time, user A might be asked to allow the app to transfer her phone number. If she accepted, the next time she wanted to download a new sound, we might ask her for her SMS messages.

As the first decision might influence consecutive ones, Table 2 contains separate statistics for the first decision users have made (first row) and the later data requests (second row). The column *Reject* shows how many users rejected a data request, *Accept* shows how many users accepted a request and *Shift of opinion* shows how many users shifted their opinion to *accept* after first rejecting a certain data request. When looking at the numbers, it is interesting to note that 59.6 % of all users trying to add a new sound finally rejected the data request. Thus, more than half of all respondents did not stick with the coarse-grained permissions they granted during installation, but reversed their decision. Only 51 users directly accepted data transfer when prompted during run-time. This provides first evidence that fine-grained data requests at the time of data access might better reflect a user's intention than coarse-grained permission requests during application installation. By displaying fine-grained information during runtime, users' likelihood to disclose their information seems to be substantially lowered (H1).

**Table 2. Descriptive statistics**

| **Decisions included** | **Reject** | **Accept** | **Shift of opinion** |
|---|---|---|---|
| First data request | 115 (59.6 %) | 51 (26.4 %) | 27 (14.0 %) |
| Later data requests | 34 (11.0 %) | 260 (84.1 %) | 15 (4.9 %) |

Interestingly, the 78 users (51 + 27) who had accepted the first data request, finally accepted further data requests in 89 % (84,1% + 4,9%) of the cases as well. As a result, the ratio of rejects to accepts was reversed for consecutive decisions (second row of Table 2). Only 34 data requests were rejected, while 275 (260 + 15) data requests were confirmed.

We tested for the effect of the level of detail on the users' acceptance using contingency analysis. No significant effect of the level of detail on user acceptance could be found when only considering the users' *first* decision ($\chi^2(1, N = 193) = .\ 1,\ p > .1$). However, when considering consecutive decisions of the users who had already accepted one of the requests, a significant effect of the level of detail could be shown. The corresponding contingency table is displayed in Table 3. For data requests which displayed fine-grained user-related information, the percentage of rejects was significantly higher than for abstract data requests ($\chi^2(1, N = 309) = 4.75$, $p < .05$). Thus, for users who had already accepted a data request, H2 could be confirmed for consecutive decisions.

**Table 3. Contingency table for later data requests**

| | **Reject** | **Accept** | **Total** |
|---|---|---|---|
| **Abstract data requests** | 20 (11.8 %) | 150 (88.2 %) | 170 |
| **Fine-grained data requests** | 29 (20.9 %) | 110 (79.1 %) | 139 |

The second research question focuses on how users' security background would influence user decisions. To investigate this influence, we developed the operationalization of users' security awareness described above and tested its effect on information disclosure. Users' security awareness ranged from .05 to .56 (mean = .25; SD = .11). Thus, all users downloading the app displayed a low or medium level of security awareness. To test for the effect of security awareness on information disclosure, we conducted binary logistic regressions for the first decisions users' made as well as for consecutive decisions. Separate regressions were necessary as the first decision might have influenced later ones. In the regression we included security awareness and level of detail as independent variables and acceptance as dependent variable. In addition, we controlled for information type, data transfer to third parties and reasoning.

Interestingly, only considering the users' first decisions in the regression, the model was not significant ($\chi^2(3, N = 193) = .11$, $p > .1$) and security awareness had no effect on users ($p > .1$). Thus, looking at the overall sample, H3 could not be confirmed. However, investigating consecutive decisions of the remaining users, the regression model was significant ($\chi^2(3, N = 309) = 17.83$, $p < .0005$). An increase in security awareness significantly reduced a user's likelihood to accept a request ($p < .05$). Thus, H3 could be partially confirmed. Further, we found a significant interaction effect of detail and security awareness on acceptance ($p < .1$) partially confirming H4 as well.

## 6. Discussion

The presented study provides several theoretical and practical contributions. For one, the study tested how fine-grained permissions influence information disclosure. As illustrated in the results section, many users rejected the data requests within the app even though they had granted the corresponding permissions already during installation. This may be due to the fact that they did not understand the permission requests in the first place, as suggested by previous research on permission dialogs that show that these dialogs are not effective in transmitting privacy information [4]. The more fine-grained data requests presented at the time of information access appear to better enable users to decide about information disclosure. Indeed, more than half of all users rejected the first data request displayed to them. This finding supports previous research that has shown that the explicitness of a permission request could influence perceived risks [42].

In contrast to Android, in iOS users do not have to grant permissions before installing the application. Instead, they are asked to confirm the permission when the app accesses (sensitive) resource information during runtime. For instance, when a restaurant finder application wants to access location data of the user, the user is shown a prompt right before location data is accessed and can grant or deny the request. This is an approach similar to our design, except for the fact that the user is shown only coarse-grained, abstract information, for instance on where the data is sent to.

In the next major Android release, Android M, Android will roll-out a permission system similar to Apple's iOS, where the user gets prompted right before the app tries to access (sensitive) resource data [43]. Our results have confirmed that from a usability point of view Apple's and Android M's permission model is a better way to design a proper permission model. However, we have further shown that more fine-grained permission messages do influence the users' decision because it gives them a better understanding of what the app does with their personal data. Such fine-grained information about data flows can positively improve the protection of the users.

Surprisingly, displaying concrete user-related information and thus, further increasing the level of detail, did not have any influence on users during their first decision about disclosure within our experiment. It seems that the sole existence of the privacy alert pointing out specific data types to be transferred at a specific point in time was enough to make nearly 60 % of users reject the request. The display of user-related information could only be shown to significantly influence users who had accepted a request before. In consecutive decisions, the display of user-related information made more users reject requests than an abstract statement. Even though H2 could be confirmed only partially we believe that our findings represent an important first step in identifying the role of direct, user-related data requests.

The effect of the level of detail on information disclosure was shown to be moderated by the users' security awareness. Only users with a medium security awareness accepted more detailed information requests. (Users with a high awareness had not installed the app in the first place.) Users with low security awareness and who had consented once generally accepted future decisions as well. This indicates that user-related, clear information communication only helps when the user at least has some awareness of privacy and security issues, but at the same time, is generally open to revealing at least some kind of data.

## 7. Limitations and Outlook

As any study, our research underlies several limitations which provide potential for future work.

For one, we offered our app in Google Play and embedded our experiment into the app. This allowed us to measure real behavior. However, at the same time this procedure deprived us of the opportunity to obtain additional information about the users, as would have been possible in an online survey. For instance, it would have been interesting to learn more about the users' perceived risk and benefits, or privacy concerns, and to test their impact on real behavior. Future work might combine a real life experiment with laboratory experiments in order to be able to compare the results and thus, gain further understanding of information disclosure.

In our experiment, we focused on one specific type of app, namely a fun application. As argued in the paper, this was appropriate as it allowed us to ensure that permissions were not associated to functionality and as this app category is especially prone to fraud apps. However, it would be interesting to explore our findings in other contexts as well. For instance, it might be possible that users would assess the requests differently if they would obtain huge perceived benefits from information disclosure, or if they had more trust in the provider.

To generate information requests utilizing users' information, we had to request all permissions necessary to access all the information that might be requested during app usage. This resulted in a list of twelve total permissions. As we offered a fun app and as it was clear that all these permissions were not necessary for the application to work, privacy sensitive users might not have installed the application. Thus, our sample is comprised of users who either do not pay close attention to permission requests or do not understand the requests or do not value privacy very much. Future studies might consider this limitation and try to find other configurations in which also privacy sensitive users might participate.

## 8. Conclusion

In our study, we combined the expertise from the fields of security and Information Systems research to determine appropriate means for communicating information access. We find that fine-grained permission requests during run-time better inform users than coarse-grained ones before installation. Further, we could show that users' security awareness and the level of detail of the data request can influence users' disclosure behavior.

## 9. Acknowledgements

This work was supported by the BMBF within EC SPRIDE and ZertApps, by the Hessian LOEWE excellence initiative within CASED, by the DFG within RUNSECURE, the Collaborative Research Center CROSSING and the Priority Programme 1496 “Reliably Secure Software Systems – RS3”.

## 10. References

[1] http://www.appbrain.com/stats/number-of-android-apps, accessed 2015-06-12.
[2] Zhou, Y., and Jiang, X., "Dissecting Android Malware: Characterization and Evolution", Proceedings of the 2012 IEEE Symposium on Security and Privacy, 2012, pp. 951-909.
[3] Rasthofer, S., Asrar, I., Huber, S., and Bodden, E., "How Current Android Malware Seeks to Evade Automated Code Analysis ", Proceedings of the 9th International Conference on Information Security Theory and Practice 2015.
[4] Felt, A.P., Ha, E., Egelman, S., Haney, A., Chin, E., and Wagner, D., "Android Permissions: User Attention, Comprehension, and Behavior", Proceedings of the Eighth Symposium on Usable Privacy and Security, 2012.
[5] Felt, A.P., Egelman, S., and Wagner, D., "I've Got 99 Problems, but Vibration Ain't One: A Survey of Smartphone Users' Concerns", Proceedings of the second ACM workshop on Security and privacy in smartphones and mobile devices, 2012, pp. 33-44.
[6] Acquisti, A., "Privacy in Electronic Commerce and the Economics of Immediate Gratification", Proceedings of the 5th ACM conference on Electronic commerce, 2004, pp. 21-29.
[7] Acquisti, A., and Grossklags, J., "Losses, Gains, and Hyperbolic Discounting: An Experimental Approach to Information Security Attitudes and Behavior", in (Camp, L.J., and Lewis, S.): The Economics of Information Security, Kluwer, 2004, pp. 165-178.
[8] Smith, H.J., Dinev, T., and Xu, H., "Information Privacy Research: An Interdisciplinary Review", MIS Quarterly, 35(4), 2011, pp. 989-1016.
[9] Norberg, P.A., Horne, D.R., and Horne, D.A., "The Privacy Paradox: Personal Information Disclosure Intentions Versus Behaviors", Journal of Consumer Affairs, 41(1), 2007, pp. 100-126.
[10] Acquisti, A., and Grossklags, J., "Privacy and Rationality in Individual Decision Making", IEEE Security and Privacy, 3(1), 2005, pp. 26-33.
[11] Conti, M., Nguyen, V.T.N., and Crispo, B., "Crepe: Context-Related Policy Enforcement for Android", Proceedings of the 13th International Conference on Information Security, 2011.
[12] Nauman, M., Khan, S., and Zhang, X., "Apex: Extending Android Permission Model and Enforcement with User-Defined Runtime Constraints", Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, 2010, pp. 328-332.
[13] Rasthofer, S., Arzt, S., Lovat, E., and Bodden, E., "Droidforce: Enforcing Complex, Data-Centric, System-Wide Policies in Android", Proceedings of the 9th International Conference on Availability, Reliability and Security, 2014.
[14] Arzt, S., Rasthofer, S., Fritz, C., Bodden, E., Bartel, A., Klein, J., Le Traon, Y., and Octeau, D., "Flowdroid: Precise Context, Flow, Field, Object-Sensitive and Lifecycle-Aware Taint Analysis for Android Apps ", Proceedings of the 35th ACM SIGPLAN Conference on Programming Language Design and Implementation 2014.
[15] Enck, W., Gilbert, P., Chun, B.-G., P., L., Jung, J., Mcdaniel, P., and Sheth, A.N., "Taintdroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones", Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation, 2010.
[16] Lu, L., Li, Z., Wu, Z., Lee, W., and Jiang, G., "Chex: Statically Vetting Android Apps for Component Hijacking Vulnerabilities", Proceedings of the 2012 ACM Conference on Computer and Communications Security, 2012.
[17] Fahl, S., Harbach, M., Muders, T., Baumgärtner, L., Freisleben, B., and Smith, M., "Why Eve and Mallory Love Android: An Analysis of Android Ssl (in)Security", Proceedings of the 2012 ACM conference on Computer and communications, 2012

[18] Sounthiraraj, D., Sahs, J., Greenwood, G., Lin, Z., and Khan, L., "Smv-Hunter: Large Scale, Automated Detection of SSL/Tls Man-in-the-Middle Vulnerabilities in Android Apps", Proceedings of the 21st Annual Network and Distributed System Security Symposium 2014.

[19] Vance, A., Anderson, B.B., Kirwan, C.B., and Eargle, D., "Using Measures of Risk Perception to Predict Information Security Behavior: Insights from Electroenecphalography", Journal of the Association for Information Systems, 15, 2014, pp. 679-722.

[20] Kahneman, D., and Tversky, A., "Prospect Theory: An Analysis of Decision under Risk", Econometrica, 47(2), 1979, pp. 263-291.

[21] Dinev, T., and Hart, P., "An Extended Privacy Calculus Model for E-Commerce Transactions", Information Systems Research, 17(1), 2006, pp. 61-80.

[22] Keith, M.J., Thompson, S.C., Hale, J., E. Hale, and Greer, C., "Examining the Rationality of Information Disclosure through Mobile Devices", Proceedings of the 33rd International Conference on Information Systems, 2012.

[23] Anderson, C., Free - the Future of a Radical Price, Hyperion, New York, 2009.

[24] Eling, N., Krasnova, H., Widjaja, T., and Buxmann, P., "Will You Accept an App? Empirical Investigation of the Decisional Calculus Behind the Adoption of Applications on Facebook", 34th International Conference on Information Systems, 2013.

[25] Krasnova, H., Eling, N., Abramova, O., and Buxmann, P., "Dangers of 'Facebook Login' for Mobile Apps: Is There a Price Tag for Social Information?", Proceedings of the 35th International Conference on Information Systems, 2014.

[26] Li, Y., "Empirical Studies on Online Information Privacy Concerns: Literature Review and an Integrative Framework", Communications of the Association for Information Systems, 28(2011, pp. 453-496.

[27] Jensen, C., Potts, C., and Jensen, C., "Privacy Practices of Internet Users: Self-Reports Versus Observed Behavior", International Journal of Human-Computer Studies, 63(1-2), 2005, pp. 203-227.

[28] Wang, N., Wisniewski, P., Xu, H., and Grossklags, J., "Designing the Default Privacy Settings for Facebook Applications", 17th ACM Conference on Computer Supported Cooperative Work & Social Computing, 2014, pp. 249-252.

[29] http://www.statisticbrain.com/mobile-phone-app-store-statistics/, accessed 2015-06-12.

[30] https://blog.avast.com/2015/02/03/apps-on-google-play-pose-as-games-and-infect-millions-of-users-with-adware/, accessed 2015-06-12.

[31] Mcafee Labs, "Mcafee Threats Report: Second Quarter 2013", McAfee,, 2013.

[32] Zhou, T., "An Empirical Examination of User Adoption of Location-Based Services", Electronic Commerce Research, 13(1), 2013, pp. 25-39.

[33] Xu, H., Teo, H.-H., and Tan, B.C.Y., "Predicting the Adoption of Location-Based Services: The Role of Trust and Perceived Privacy Risk", 26th International Conference on Information Systems, 2005.

[34] Petrova, K., and Wang, B., "Location-Based Services Deployment and Demand: A Roadmap Model", Electronic Commerce Research, 11(1), 2011, pp. 5-29.

[35] Keith, M.J., Babb, J.S., Furner, C.P., and Abdullat, A., "Privacy Assurance and Network Effects in the Adoption of Location-Based Services: An Iphone Experiment", Proceedings of the 31st International Conference on Information Systems, 2010.

[36] Xu, H., Teo, H.-H., Tan, B.C.Y., and Agarwal, R., "The Role of Push-Pull Technology in Privacy Calculus: The Case of Location-Based Services", Journal of Management Information Systems, 26(3), 2009, pp. 135-173.

[37] Krasnova, H., Eling, N., Schneider, O., Wenninger, H., Widjaja, T., and Buxmann, P., "Does This App Ask for Too Much Data? The Role of Privacy Perceptions in User Behavior Towards Facebook Applications and Permission Dialogs.", 21st European Conference on Information Systems, 2013.

[38] Svajcer, V., "Sophos Mobile Security Threat Report", SOPHOS, 2014.

[39] Kelley, P.G., Consolvo, S., Cranor, L.F., Jung, J., Sadeh, N., and Wetherall, D., "A Conundrum of Permissions: Installing Applications on an Android Smartphone", in (Blyth, J., Dietrich, S., and Camp, L.J.): Financial Cryptography and Data Security, Springer Berlin Heidelberg, 2012, pp. 68-79.

[40] Aviv, A.J., Gibson, K., Mossop, E., Blaze, M., and Smith, J.M., "Smudge Attacks on Smartphone Touch Screens", Proceedings of the 4th USENIX Conference on Offensive Technologies, 2010.

[41] http://www.androidauthority.com/android-jelly-bean-face-unlock-blink-hacking-105556/, accessed 2015-06-12.

[42] Bal, G., "Explicitness of Consequence Information in Privacy Warnings: Experimentally Investigating the Effects on Perceived Risk, Trust, and Privacy Information Quality", Proceedings of the 35th International Conference on Information Systems, 2014.

[43] https://developer.android.com/preview/features/runtime-permissions.html, accessed 2015-06-12.