A Comparative Study of Android Users' Privacy Preferences under the Runtime Permission Model

Panagiotis Andriotis[1], Shancang Li[1], Theodoros Spyridopoulos[1], and Gianluca Stringhini[2]

[1] Cyber Security Research Unit,
University of the West of England, Bristol, BS16 1QY, U.K.
panagiotis.andriotis@uwe.ac.uk
[2] University College London, London, WC1E 6BT, U.K.

Abstract. Android users recently were given the ability to selectively grant access to sensitive resources of their mobile devices when apps request them at runtime. The Android fine-grained runtime permission model has been gracefully accepted by the majority of users, who also seem to be consistent regarding their privacy and security preferences. In this paper we analyse permission data collected by Android devices that were utilising the runtime permission model. The reconstructed data represent apps' settings snapshots. We compare behavioural insights extracted from the acquired data with users' privacy preferences reported in our previous work. In addition, compared with the responses received from another group of mobile device users, users' privacy settings seem to be affected by the functionality of apps. Furthermore, we advise visual schemata that describe users' privacy settings and point out a usability issue regarding the installation process of Android apps under the runtime permission model.

Keywords: Runtime, Permissions, Android, Settings, Apps, Marshmallow, Privacy, Security, Usability, Profile

1 Introduction

Mobile device sales are increasing every year [1] and consequently, more people are nowadays able to use smartphones and tablets. Contemporary portable devices are usually equipped with numerous sensors and advanced storage capacity. Hence, provided software (apps) can nowadays perform complex tasks and, therefore, produce large amounts of highly personalised data. This type of pervasive technology has been widely criticised in the past, with criticism focusing particularly on the privacy issues it accumulates [8].

Until recently (i.e., autumn 2015), Android users could not control which resources should be available to apps during runtime. However, modern versions of the most popular operating systems (Android and iOS) are now equipped with permission control management systems. One common characteristic of

these systems is that when an app needs to access for the first time specific components of the system, e.g. the microphone, the user has to explicitly grant (or deny) permission for this action. Thus, users' privacy and security awareness can be, theoretically, increased.

We can categorise Android versions in two generations, regarding their permission system. The old generation devices (until version 5.1.1) do not allow users to permit or deny access to sensitive resources during runtime, when apps make such requests. The permission model used on devices that run old generation versions requires the user to accept, before installation, all permissions an app might request. In other words, when users want to install an app via the Google Play app store, which is the official Android marketplace, they will see a list of requested permissions before they install the app. Then they have two choices; either accept and continue the installation process (granting thus access to all requested resources), or cancel the installation. Users of the new generation versions might also see a similar list of requested permissions before installing an app. Then, they also need to accept or deny the app installation. The major difference with the old generation versions is that, if an app requests access for the first time to sensitive resources (during runtime), the system will issue a dialogue message requesting from the user to deny or grant access to the specific resource. According to the Android Developer documentation [2], as of February 2017 there exist nine groups of dangerous permissions: Calendar, Camera, Contacts, Location, Microphone, Phone, Sensors, SMS, Storage.

Runtime permissions are increasing users' security and privacy awareness and, in theory, allow them to handle more efficiently the personal data they share. Users can revoke granted permissions anytime using the *Settings* app. Thus, if for example we assume that users do not prefer to share their SMS list with other ecosystems (i.e., apps), they are given the ability under the current permission model to install apps that function properly, but at the same time, they cannot access restricted areas of the device; these areas are set by the users.

We recently conducted a study aiming to examine how Android users adopted to this change [4]. The study demonstrated that users make in general consistent choices regarding the resources they are more willing to grant access. In this paper, we compare previous results with the analysis of a new dataset that came from a different group of Android users. We report similar trends on the way these two different user groups handle permission requests from social (and messaging) apps. Additionally, the current study demonstrates that although people have specific perceptions when asked which resources would be more reluctant to allow an app to access, they eventually overlook their priorities when they need to benefit from functionalities that various apps offer. Also, we stress the luck of sufficient information on the official marketplace regarding the *least required permissions*, which are needed in order an app to function properly. Finally, we conclude this paper by proposing visual schemes that could be used to represent users security profiles.

The rest of this paper is organised as follows: we discuss related work in the next Section and present the methodology we used to gain our dataset in Sec-

tion 3. Section 4 demonstrates our data analysis method and Section 5 presents our results. We further discuss the outcomes and limitations of our work in Section 6 and conclude this paper in Section 7.

2 Related work

Over-privileged applications introduce security threats to mobile device ecosystems and pose various reputational risks to online markets such as the Android marketplace [13]. Such threats, according to [13], often derive from the use of advertising libraries. Additionally, users are often not aware of the context of the permissions they granted to installed applications in the past [3]. As a matter of fact, a recent study demonstrated that the majority of users would prevent at least one requested permission from an experimental application, if they knew beforehand the purpose of this request [14]. Furthermore, mobile device users are often astounded by the capabilities of various apps to collect personal data and share them with third-party entities [11]. If an app, for example, is able to gain access to personal data, such as the device's list of incoming or outbound SMS, it is even possible to acquire information about the emotional state of the entities that exchange these messages [5].

Consequently, previous research studies proposed extension mechanisms that would allow to overcome privacy and security constrains of the old permission model [9]. Researchers also introduced in the past fine-grained access control methodologies for Android apps using explicit policies [12]. FlaskDroid [9] for example, offers a flexible fine-grained access control system. Other systems were designed to protect only specific streams of data, such as users' location [7]. A popular technique used for location data protection is obfuscation [10]. The concept of obfuscation is simple; the system feeds with shadow or fake data any application that requests access to location services [6].

Fine-grained access control has been introduced to mobile device users in the past, when the iOS 6 was launched. However, Android users became familiar with the runtime permission model when the sixth version was presented (Marshmallow). In our previous work [4], we presented the first study that was focused on Android Marshmallow users. The study suggested that Android users were comfortable with the runtime permission model. Moreover, data derived from 50 participants demonstrated that Android users make consistent choices regarding their privacy preferences. In the current work we aim to compare responses and permission data from a different group of users with the previous study.

3 Methodology

We developed an Android app, targeting users of the 'Marshmallow' or 'Nougat' versions. In order to apply this limitation, the app was compiled utilising the `android:minSdkVersion` attribute of the `AndroidManifest.xml` file of the app; this attribute was set to API level 23 (Marshmallow). This app was used in our previous work [4], but for the needs of the current experiments we included

additional fractions of code to get the timestamps of the first installation of each installed app and its latest update. At a later version, we also included in the collected data the `targetSDKVersion` attribute of each app, aiming to figure out if the installed apps in each device were compiled following the standards of the newest Android version (currently API 25, as on February 2017).

Participants could download the app via the official Android app marketplace (Google Play); the app was named *"Permissions Snapshot"*. After installing the app, participants had to launch our survey-app and read the Information Sheet and Consent Agreement. If they were satisfied with the terms of use, they had to click on a check box; then the data collection procedure succeeded. The respondents were asked to provide some basic demographic data and then they had to answer 6 questions related to the use of the runtime permission model. As described in our previous work [4], the app was also collecting data about the permissions that were granted to the installed apps at the given time. This was achieved utilising the `PackageManager` class. At the end of this procedure the participants had to upload to a server the collected data by clicking a button. They also had the opportunity to read a very brief tutorial, which reminded them that they could check and control app permissions anytime, using the *Settings* app of their devices.

The questions (also described in our previous work [4]) can be summarised as follows: The participants were asked: 1) how long they were using the current Android version; 2) if they had noticed any changes at the permission model; 3) if they believed that using the runtime model they had gained additional control over the data they were sharing; 4) if they knew that they could grant, deny or revoke permissions using the Settings app; 5) if they found the runtime permission model irritating, "because it asks too many questions" and 6) if they prefer the runtime permissions model or the previous one. Note that this study was approved by the UCL Research Ethics Committee *(Project ID Number: 8945/001)*.

As a second step to our experiments, we were aiming to highlight how users would probably react when an app requested access to sensitive resources of their devices. Without a doubt, the act of granting permissions to apps to allow them to access specific resources of a mobile device, relates primarily to the apps' functionality. However, it is also interesting to see how users prioritise the resources they consider as more sensitive. To this end, we asked the following question to 4 groups of undergraduate students, using an online tool *("poll everywhere")*: "Assume an Android app requests access to your phone's resources. Rank them according to the possibility to allow the app to access them. On the top of the stack you should place the group that you are more keen to permit access: Calendar, Camera, Contacts, Location, Microphone, Phone, Sensors, SMS, Storage". The students were given five minutes to answer the question. They could rearrange the groups of dangerous permissions using the interface provided by the online tool. The groups were shown as a stack and the participants could use the mouse and rearrange the stack. We eventually collected anonymous responses from 25 unique participants. We refer to this group of participants as the "on-

line questionnaire respondents". The results of this experiment are presented at Section 5.

4 Data Collection and Analysis

"Permissions Snapshot" collected anonymous permission data from participants' mobile devices. The app was using the `PackageManager` class to accumulate information about the installed apps (packages) on each device. We made use of its `getInstalledApplications` method with the `GET_META_DATA` flag to acquire access to the packages; then we invoked the method `getPackageInfo` with the `GET_PERMISSIONS` flag to get information about the packages. The requested permissions were acquired using the `requestedPermissions` and `requestedPermissionsFlags` attributes of the `PackageInfo` data types. The `PackageInfo` data type also returned information about the first installation time, the last update time, and the target SDK version of each installed app, i.e., `firstInstallTime`, `lastUpdateTime`, `targetSdkVersion`. Note that we excluded system apps from the data collection process.

Granted permissions for each installed app were obtained using the correlation of data received from the following data types: `packageInfo.requestedPermissions` and `packageInfo.requestedPermissionsFlags`. The former is an array of `String`s and the latter is an array of `int`s. These data types describe users' settings related to the permissions granted to each app at the given time. Assuming that we are interested to see if a user granted permission to the Facebook app to access the Camera group, we need to see which flag *(int)* is associated with the permission `android.permission.CAMERA`. As suggested in [4], when this flag is equal to 1 this means that the permission was not granted. On the other hand, if the permission was granted, then this number would be equal to 3.

The permission data we analysed and present in this paper are complimentary to the data presented in our previous work [4]. They were collected from 13 participants using the same app ("Permissions Snapshot") that was used in [4]. These participants do not belong to the group of Android users that participated in our previous work. Also, the responses presented in this paper were basically collected during the last three months of 2016. Users' anonymity was maintained by calculating a hashed value of the `ANDROID_ID` of each device. This hexadecimal number describes uniquely a device; we used a hashed value of it to conceal the real identity of the user and, at the same time, to avoid having duplicate entries from the same device.

The collected responses from the 13 participants also contained demographic data and the answers to the aforementioned questions (Section 3). The participants provided basic demographics (Gender, Age, Area of Residence) and then were asked to answer six multiple-choice questions. Each question was presented sequentially having a predefined answer to allow us identify any users that where just skipping the questions by clicking the "Next" button. Two responses contained the predefined answers to all multiple-choice questions and were thus excluded from this presentation. However, their permission data were included in our analysis since users cannot manipulate them; these are device-dependent data. Moreover, two files did not contain demographic data; they only contained permission data. Hence, the following demographic information was extracted from nine participants.

Most of the respondents were males (88.88%) and they were between 18 to 30 years old (77.78%) or 31 to 46 years old (22.22%). We got valid responses from Europe (66.67%), Asia (22.22%) and the Americas (11.11%). Additionally, one third of the

participants claimed they were using the current Android version for 0 - 6 months, one third said they were using it for 7 - 12 months, 22.22% chose the "More than 1 year" option and 11.11% selected "I Dont Know". According to the replies to the second question, most participants (77.78%) had noticed the changes to the permission model. Also, the majority of users (88.89%) agreed that under the new permission model they felt they could control the data they share more efficiently. In the fourth question, 66.67% of the participants knew that they could revoke, or grant access to installed apps using the *Settings* app of their devices (tapped the "Correct" option); 22.22% chose the "Wrong" option and 11.11% replied "I Dont Know". Furthermore, 55.56% suggested that they were not frustrated by the fact that the system interacts with the user frequently, asking them to grant permissions. Finally, users' preference for the runtime model is evident (77.78%). The rest 11.11% preferred the previous model and 11.11% chose the "I don't have any preference" option.

Compared to the responses from our previous work, we identify similarities in most cases. The only difference we noticed here (in question 5) is that 44.46% of the respondents believed that under the runtime permission model, the system interacts very frequently with the users, requesting from them to grant permissions to the app that runs at the foreground. This is quite interesting considering the fact that in our previous study only 15% of the respondents expressed the same belief. However, the responses demonstrate that the participants were knowledgeable and felt they have more control on the data they share under the runtime permission model. Thus, they eventually prefer the runtime permission model against the old one.

Considering the replies from the group of 25 respondents, who prioritised the dangerous groups according to the possibility to allow an app to access them, we can see that these users are keener to allow access to the Calendar or the Sensors of their devices and are more hesitant to grant permission to the SMS or the Microphone groups. The dangerous groups that participants would be more willing to allow an app to access (in descending order) are as follows: Calendar, Sensors, Storage, Location, Contacts, Camera, Phone, SMS, Microphone. In the next Section we will further discuss these responses. The next Section presents results obtained from the analysis of the collected permission data from 13 Android devices.

5 Results

The results presented in this Section derive from data received from 13 unique Android devices. Note that this sample might not be large enough to provide clues about particular behavioural characteristics of Android users. However, in this paper, we refer to this group of users in order to compare previous results and eventually confirm our findings, presented in [4].

5.1 General Information

The devices of the 13 participants contained a variety of data. On average, 46 apps were installed in each device (we do not consider system apps in this study). This is a rough estimation because there were some outliers in our sample. For example, device No4 contained 197 apps and, on the other hand, devices No5 and No9 contained only 4 and 2 apps, respectively. If we do not consider the latter, the average number is 54 apps per device. The participants' devices contained 523 unique apps. As already

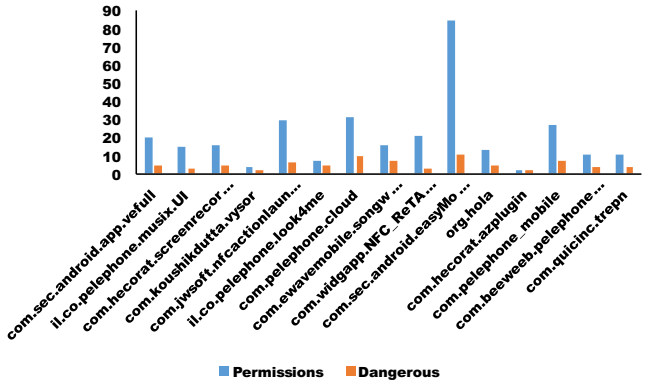


Fig. 1. Number of declared permissions (blue colour) and dangerous permissions (orange colour) in installed apps on device No1.

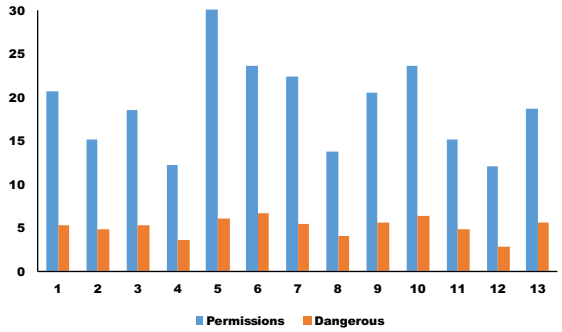


Fig. 2. Average number of declared permissions (blue colour) and dangerous permissions (orange colour) per device.

mentioned, the maximum number of installed apps on a device was 197 and the minimum was only 2. The app that declared the maximum number of permissions (83) is the tool `com.sec.android.easyMover`, which transfers data across different devices. Moreover, `com.quickheal.platform` and `org.thoughtcrime.securesms` were the apps that declared the maximum number of dangerous permissions (20).

Figure 1 shows the number of permissions declared in each installed app on device No1. In addition, the same figure shows the number of dangerous permissions included in these apps. Roughly, one can estimate that almost one third of the declared permissions per app belong to dangerous groups. Furthermore, figure 2 demonstrates the average number of declared and the average number of dangerous permissions per device. Again, the same trend is evident in the specific figure. Finally, we calculated the declared and the dangerous permissions of the apps that constitute our sample. The average number of declared permissions per app is 15.92 and the average number of dangerous permissions is 4.61. This means that 28.96% of the declared permissions in

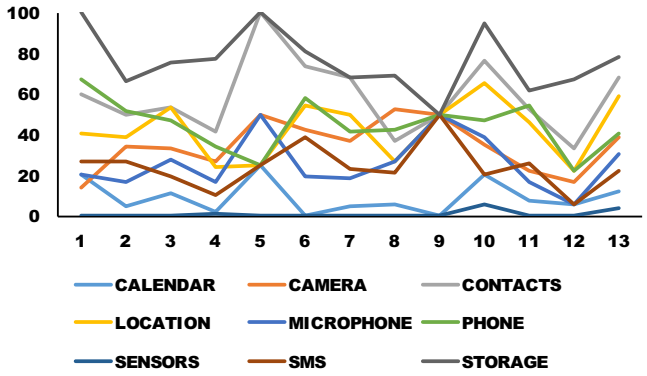


Fig. 3. Percentage of apps per device that request access to sensitive resources.

Table 1. Most requested dangerous permission groups by installed apps (on average).

Permission groups	Our study (%)	Previous work [4] (%)
CALENDAR	9.05	7.34
CAMERA	34.71	30.46
CONTACTS	58.68	58.02
LOCATION	42.42	49.67
MICROPHONE	25.78	21.63
PHONE	44.45	40.56
SENSORS	0.75	0.93
SMS	24.12	16.51
STORAGE	75.82	76.64

each device belongs to dangerous groups. Compared to our previous study, where we reported (on average) approximately 12.39 declared and 3.85 dangerous permissions per app, we can see that the same trend exists in the current work; nearly 30% of the requested permissions belong to dangerous groups.

5.2 Dangerous Permission Groups

Figure 3 demonstrates the percentage of apps per device that request access to sensitive resources, according to the declared permissions in their `AndroidManifest.xml` files. The figure illustrates that most apps request access to the devices' Storage group. Contacts, Phone, Location and Camera are also the most requested resources. Table 1 shows on average which are the most requested dangerous permission groups by the installed apps. The last column indicates the results of our previous work [4]. The comparison between the two groups of participants shows that Storage, Contacts, Location or Phone, and Camera are the most requested dangerous groups per device (considering permissions requested in the `AndroidManifest.xml` file).

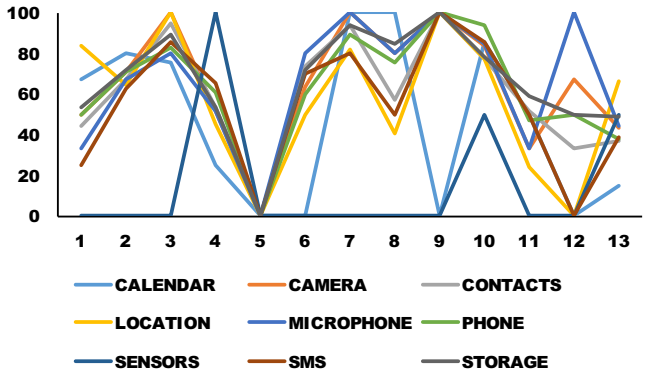


Fig. 4. Percentage of apps that were granted access to dangerous permissions per device.

Android's runtime permission model was introduced in order to make the user aware that an app needs to access resources that deemed to be sensitive. Hence, figure 3 indeed provides insightful information about the most requested dangerous groups. However, it does not actually show which permissions were granted when participants completed our survey. Thus, using information gathered from the `PackageManager` (as discussed in Section 4), we estimated which resources were open to installed apps for each device. In other words, we calculated the percentage of apps per device that seemed to have access to sensitive resources; figure 4 showcases these results. Note that the accessibility of sensitive resources for each user differs. For example, user No11 seems to be more reluctant to grant access to sensitive resources, compared to user No3. Also, it seems that most participants had a stable behaviour when granting access to sensitive resources. For instance, the accessibility rates of user No2 are between the range of 64% to 70% for most sensitive resources. However, so far we might have included in our analysis apps that were never used by the participants. This limitation occurred because it is not possible to get usage statistics from contemporary versions of the Android OS, without having users' permission.

Another limitation of the current study (which was also reported in [4]) is related to the fact that if we utilise the `PackageManager` to get permission information for apps that were compiled with parameter "targetAPIVersion" < 23, then the `packageInfo.requestedPermissionsFlags` variables will always return the integer 3. This means that we consider that permissions were granted by default to all dangerous groups for these specific applications.

5.3 Fine-grained Permissions

To overcome the former limitation, we continued our analysis considering data generated from apps that contained only fine-grained user preferences. Thus, for the rest of this section we consider fine-grained permission data from the following users: 2, 3, 4, 6, 10, 11, 13. The numbers of apps with fine-grained permission settings considered from each device respectively are: 12, 6, 11, 7, 6, 6, 22. We will refer to this group as the *"f-g" group*.

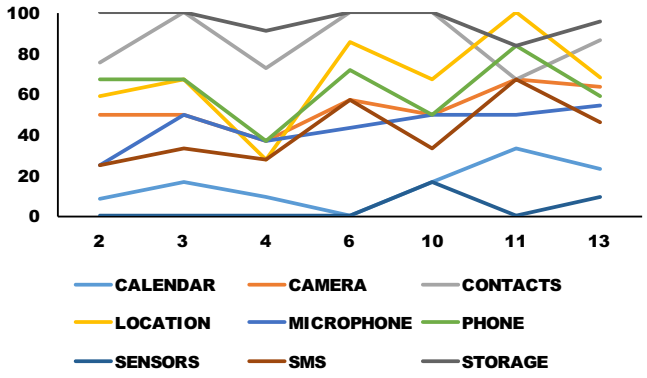


Fig. 5. Percentage of apps in the *f-g* group that requested access to sensitive resources.

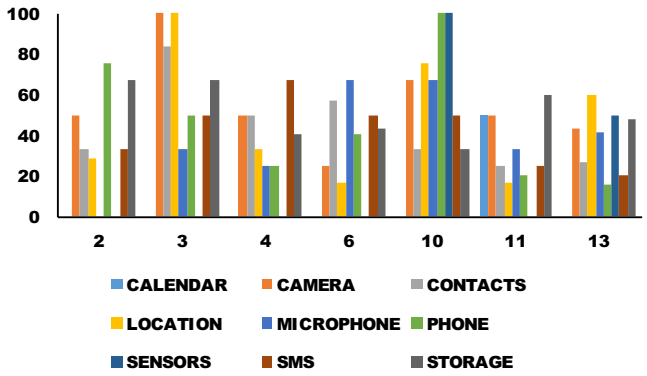


Fig. 6. Percentage of accessible resources per device in the *f-g* group.

Figure 5 shows the percentage of apps in the f-g group that requested access to sensitive resources. The most requested resources on average are as follows: Storage 95.67%, Contacts 85.82%, Location 67.56%, Phone 61.94%, Camera 53.40%.

Figure 6 demonstrates the accessibility to sensitive resources focusing on the participants with fine-grained permission settings. These results are also provided in table 2. Table 2 shows that the most accessible resources (in descending order) in the f-g group were: Sensors, Camera, Storage, Location, Phone, Contacts, SMS, Microphone, Calendar.

Furthermore, we compared the responses from the two groups of our study (the 'online questionnaire respondents' and the 'f-g group participants') and compiled resulted preferences in table 3. The first column lists the permissions that respondents were more willing to allow an app to access; the second column lists on average the accessibility to sensitive resources in the devices of our f-g participants, as shown in table 2 (both in descending order). Interestingly, we can see common trends in these two groups; first, Sensors appear to be more accessible in both groups. This means

Table 2. Percentage (%) of accessible resources in the *f-g* group.

Groups	No2	No3	No4	No6	No10	No11	No13	Average
CALENDAR	0	0	0	N/A	0	50.00	0	8.33
CAMERA	50.00	100.0	50.00	25.00	66.67	50.00	42.86	54.93
CONTACTS	33.33	83.33	50.00	57.14	33.33	25.00	26.32	44.06
LOCATION	28.57	100.0	33.33	16.67	75.00	16.67	60.00	47.18
MICROPHONE	0	33.33	25.00	66.67	66.67	33.33	41.67	38.09
PHONE	75.00	50.00	25.00	40.00	100.0	20.00	15.38	46.48
SENSORS	N/A	N/A	N/A	N/A	100.0	N/A	50.00	75.00
SMS	33.33	50.00	66.67	50.00	50.00	25.00	20.00	42.14
STORAGE	66.67	66.67	40.00	42.86	33.33	60.00	47.62	51.02
Average	35.86	60.42	36.25	42.62	58.33	35.00	33.76	

Table 3. Comparison between the priority list provided by questionnaire respondents and the accessibility of resources noticed in the f-g group.

Questionnaire	f-g group
CALENDAR	SENSORS
SENSORS	CAMERA
STORAGE	STORAGE
LOCATION	LOCATION
CONTACTS	PHONE
CAMERA	CONTACTS
PHONE	SMS
SMS	MICROPHONE
MICROPHONE	CALENDAR

that if an app requests access to the device's sensors, it is very possible that the user will grant this permission. In addition, SMS and Microphone groups are located at the bottom of both columns. On the contrary, Camera appears to be the second most accessible resource in the f-g group and Calendar the least accessible. These results might indicate that even though users believe they should not provide access to the Camera when requested by an app, they are more keen to do so when this request is actually made. Additionally, despite that questionnaire respondents replied they would allow an app to access their Calendar, we see that the f-g group accessibility to the Calendar resources received the lowest percentage. This phenomenon may have occurred as a result of the fact that some apps request access to the Calendar as a secondary feature of their functionality, thus users did not have the chance to grant or deny access to it.

5.4 Messaging and Social Media Apps

The latter finding urged us to focus on specific apps, aiming to identify connections between functionality and users' privacy settings. The most popular (social) apps in our

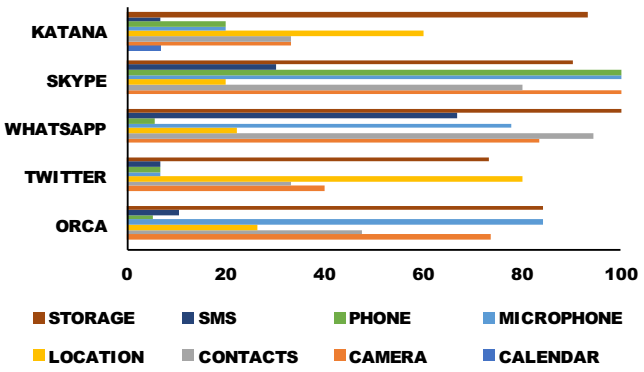


Fig. 7. Percentage of users that allowed access to sensitive resources (previous study).

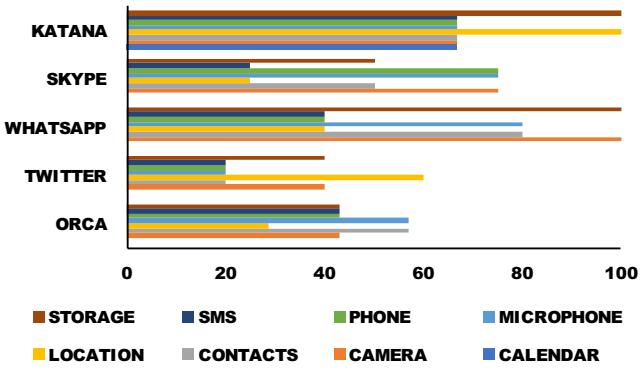


Fig. 8. Percentage of users that allowed access to sensitive resources (current study).

dataset were the following: Facebook Messenger, Twitter, WhatsApp, Skype, Facebook, Instagram, Snapchat and Slack. Given the small number of participants in this study we will only present comparative results between the current and our previous work for the following apps: Facebook Messenger, Twitter, WhatsApp, Skype, Facebook.

Figures 7 and 8 illustrate the percentage of users that allowed access to sensitive resources after the aforementioned apps requested it. First, we should notice that, in general, the rates of accessibility to the various resources seem to follow the same patterns. For example, WhatsApp users allow access to Camera, Contacts, Microphone, and Storage in both studies. Also, the rates of accessibility to the Storage group for all apps are very high in the current and our previous work. Hence, we conclude that it is very possible an app to get access to the storage of a device when such a request has been made.

Other common characteristics can be highlighted when we examine Skype users; Camera, Contacts, Microphone and Phone are the most accessible resources. Moreover, Facebook Messenger is a similar app, having the same functionality with WhatsApp

Table 4. Percentage of users that allowed access to sensitive resources (previous study).

Groups	ORCA	TWITTER	WHATSAPP	SKYPE	KATANA
CALENDAR	0	N/A	N/A	N/A	6.67
CAMERA	73.68	40.00	83.33	100.0	33.33
CONTACTS	47.37	33.33	94.44	80.00	33.33
LOCATION	26.32	80.00	22.22	20.00	60.00
MICROPHONE	84.21	6.67	77.78	100.0	20.00
PHONE	5.26	6.67	5.56	100.0	20.00
SMS	10.53	6.67	66.67	30.00	6.67
STORAGE	84.21	73.33	100.0	90.00	93.33

Table 5. Percentage of users that allowed access to sensitive resources (current study).

Groups	ORCA	TWITTER	WHATSAPP	SKYPE	KATANA
CALENDAR	0	N/A	N/A	N/A	66.67
CAMERA	42.86	40.0	100.0	75.00	66.67
CONTACTS	57.14	20.00	80.00	50.00	66.67
LOCATION	28.57	60.00	40.00	25.00	100.0
MICROPHONE	57.14	20.00	80.00	75.00	66.67
PHONE	42.86	20.00	40.00	75.00	66.67
SMS	42.86	20.00	40.00	25.00	66.67
STORAGE	42.86	40.00	100.0	50.00	100.0

and Skype. One can notice that the most accessible dangerous groups for Messenger are: Camera, Contacts, Microphone and Storage. In addition, the use of Location services was found to be popular among Twitter and Facebook users. Tables 4 and 5 show the percentage of users that allowed access to sensitive resources (considering messaging apps) in the previous and the current study, respectively.

Those common trends, derived by the two different groups of Android users, indicate that their privacy settings and preferences depend on the functionality of certain apps. Furthermore, given that users have a consistent attitude when apps request permission to access specific resources, as demonstrated in [4], it comes with no surprise the fact that we observed high accessibility to three dangerous groups (Camera, Contacts, Microphone) for three different messaging apps (Messenger, WhatsApp, Skype). Moreover, as discussed previously in this section, despite that people believe they would be more hesitant to allow access to the Camera or the Microphone groups, they eventually grant permission to specific apps with similar functionality (messaging apps). However, additional research needs to be conducted to investigate whether users' privacy preferences are related with their trust on specific apps.

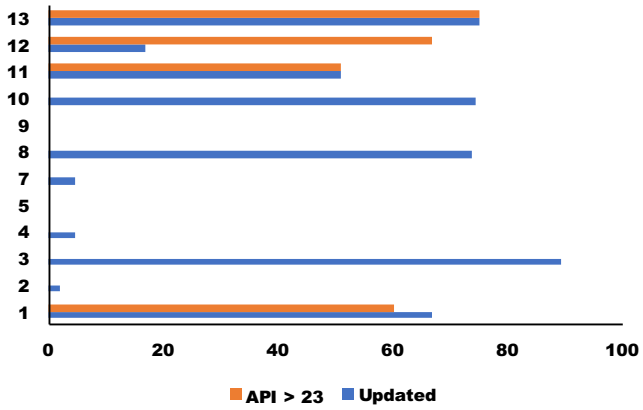


Fig. 9. Percentage of installed apps per device that had been updated at least once (blue colour) and percentage of installed apps compiled with `targetSDKVersion` > 23 (orange colour).

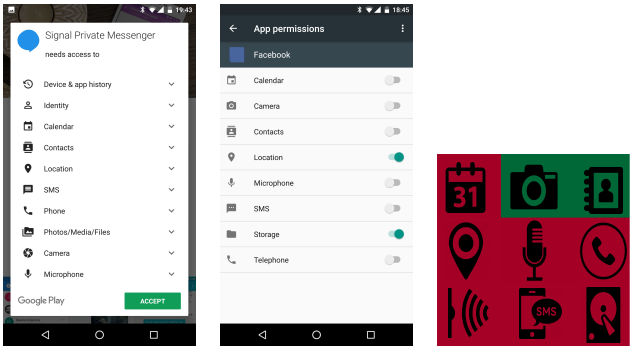
6 Discussion

Section 3 discussed the methodology we used to acquire data from participants' devices. We also mentioned that additional metadata were included in the gathered information, i.e., the timestamp of the last update of installed apps and, in some cases, the API version that was used to compile these apps. The latter information was not collected from all devices because we incorporated this feature in later versions of our app.

6.1 App Updates and API Status

Figure 9 illustrates our findings. Blue bars indicate the percentage of apps per device that had been updated at least once (either manually or automatically). Orange bars show the percentage of apps per device compiled with `targetSDKVersion` greater that API 23. Analysis showed that 50% of the participants did not update their apps or, they selectively updated a small number of apps (up to 17% of the apps were updated in some cases). Additionally, figure 9 demonstrates that 50% of the users in our sample updated at least 50% of their apps sometime in the past. Hence, we can deduct that in our small sample of users, only half of them regularly update their apps. Thus, one can suggest that 50% of the users are more vulnerable to malware, given that the updated version of an app can be considered more secure from its previous version.

Additionally, considering data collected from devices No1, No11, No12, No13 (see figure 9), we deduce that, approximately 63% of the installed apps (per device) were designed to be compatible with APIs 23, 24 and 25 (API 25 was the most modern version as of February 2017). This means that these apps were fully utilising the capabilities of the runtime permission model. Hence, we can consider that behavioural trends noticed in this study are not substantially skewed by the existence of older apps in some devices. Moreover, as metioned earlier, we did not collect app usage statistics to avoid engaging our volunteers (Android users) in going through additional steps,



(a) Permission list (b) Settings (c) LRP

Fig. 10. Screenshots showing a permission list before installation (a), granted permissions to an installed app (b), and a visual schema describing the *least required permissions* (LRP) for an app.

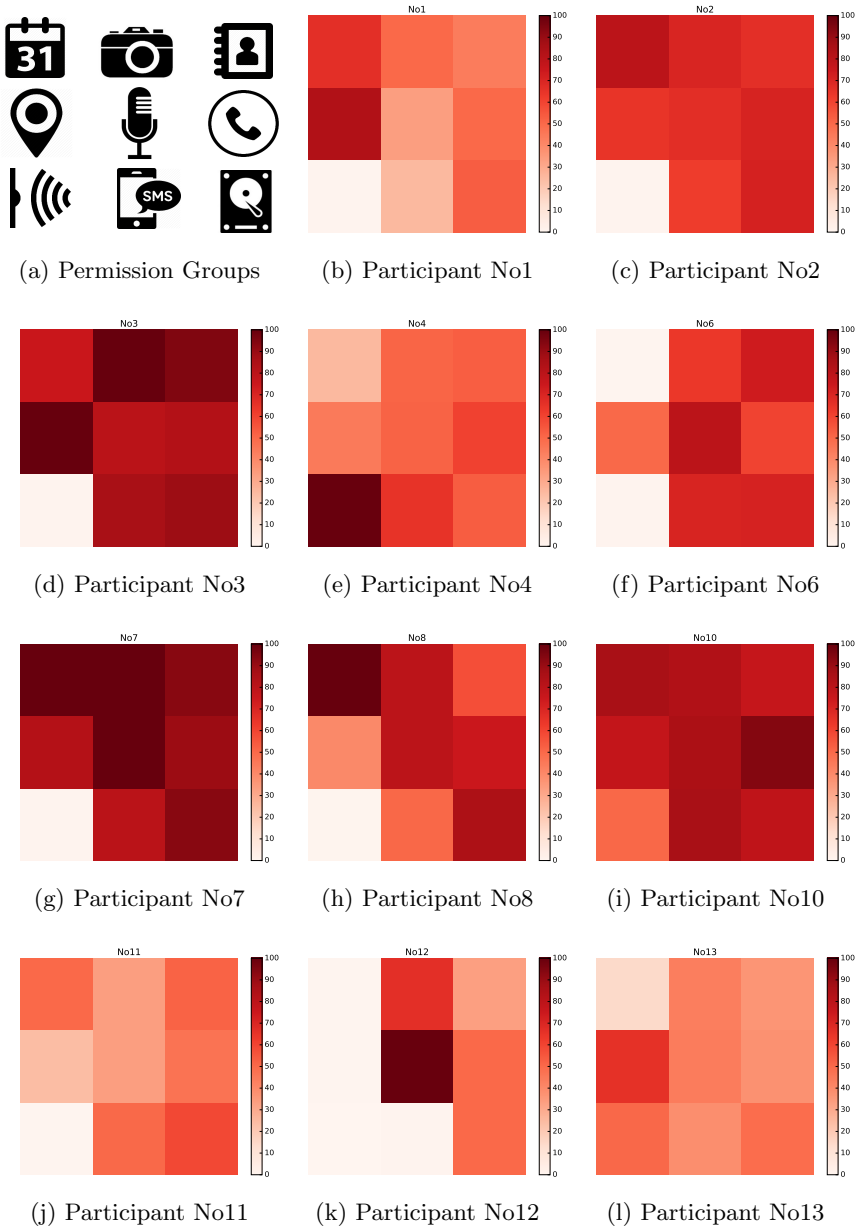
such as manually turning on the usage statistics feature for our app. Thus, results presented in Section 5.2 might contain information about apps that were never used.

6.2 Privacy Profiles

The usual (and preferred) app installation procedure on an Android device is as follows: Users navigate to the Google Play app store (or any other third-party marketplace), where they can search for their preferred app. Before downloading the app, users have the chance to see a list of permissions that might be requested (figure 10a). After installation, and during runtime, if a sensitive resource needs to be accessed, the system will issue a message dialogue to get user's approval to access the resource. Users can review and revoke the granted permissions via the Settings app (figure 10b).

Section 5.4 showcased privacy settings that seem to be common among users. For example, we noticed that all WhatsApp users allowed access to the Storage. Thus, one could suggest that the app will not be functional if permission to the Storage is not granted. However, users do not have a priori knowledge of this information before they download the particular app. Such information is not provided by the official (or third-party) app marketplaces. Hence, we can state that there exists a gap in the installation procedure, which needs to be filled. In other words, we suggest that users should be able to get basic information about the permissions they need to grant to apps in order to get the minimum functionality they can offer. For instance, the popular messaging app "Google Allo" will not function if access is not granted to (at least) the Contacts, SMS and Storage groups. Another example could be the "Google Duo" video app, which cannot function, unless the user grants access to (at least) the Camera or the Microphone group.

Google Duo is a video app and, indeed, the use of camera is obviously needed, but we believe that users would benefit from schemes that would inform them about the basic requirements of an app. In figure 10c, we propose a visual scheme that aims to concisely indicate the minimum permission requirements needed by Android apps in order to function properly. In particular, we consider that a representation of the



(a) Permission Groups (b) Participant No1 (c) Participant No2

(d) Participant No3 (e) Participant No4 (f) Participant No6

(g) Participant No7 (h) Participant No8 (i) Participant No10

(j) Participant No11 (k) Participant No12 (l) Participant No13

Fig. 11. User profiles representing the accessibility of sensitive resources on their devices.

dangerous permission categories, placed on a 3x3 grid and marked with red and green background colours according to the corresponding accessibility requirements of an app, would probably visualise sufficiently the minimum functionality requirements of this app. Figure 10c demonstrates the proposed visual scheme for an app that will not function unless access to the Camera and Contacts groups is given. Such schemes might also visually compliment (or substitute) existing system information (figure 10b).

Finally, although graphs like the one presented in figure 4 provide insightful information about the percentage of granted permissions per category (and per device), it would be also useful to propose schemes that efficiently describe these data visually. Hence, we could create a "privacy profile" that represents each user. Figure 11 presents the profiles of 11 participants. Each profile is basically a heat map which represents the percentage of apps (per device) that were granted access to the 9 categories of dangerous permission groups. For example, participant No9 is more keen to provide access to the Calendar, Camera and Microphone, compared to other participants. Moreover, we could say that users like participant No11 or No13 seem to be more cautious when granting permissions to apps, compared to users like No3, No7 or No10.

Further work needs to be done in order to investigate if users who are more keen to grant permissions to apps are more vulnerable to malware attacks. This knowledge might assist major app distributors to fight malware expansion. In addition, further work on the usability of schemes like the one presented in figure 10c needs to be done, to assess their usefulness and value to the users' experience provided by online stores.

7 Conclusions

Privacy-aware users were anticipating the advent of Android's fine-grained permission model for a long time. Our previous study showed that the majority of Android users positively adapted to this major update. This paper confirmed that most users prefer the new model. Furthermore, we confirmed that the vast majority of Android apps request access to the devices' storage and that most users are willing to permit this action. Moreover, we deduced that although people are more reluctant to allow access to resources such as their cameras or microphones, they tend to grant these permissions to specific app categories. They overcome their initial hesitation to benefit from the provided apps' functionality. We also noted that almost half of the participants in this research work did not update their apps; the other 50% of the participants seem to update their apps regularly. Furthermore, we suggested that users should be informed about the least required resources an app needs to provide its basic functionality. Hence, we proposed a visualisation scheme which could be used in online app marketplaces. Finally, this paper suggested the use of heat maps to represent users' privacy profiles. We intent to test the usability of the proposed schemes as part of future work.

Acknowledgements

This work has been supported by the EPSRC: grant number EP/N008448/1.

References

1. Global smartphone sales by operating system from 2009 to 2015 (in millions). https://www.statista.com/statistics/263445/

global-smartphone-sales-by-operating-system-since-2009/, [Online; accessed 10-February-2017]
2. Normal and Dangerous Permissions. https://developer.android.com/guide/topics/permissions/requesting.html#normal-dangerous, [Online; accessed 10-February-2017]
3. Almuhimedi, H., Schaub, F., Sadeh, N., Adjerid, I., Acquisti, A., Gluck, J., Cranor, L.F., Agarwal, Y.: 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. pp. 787–796. CHI '15, ACM, New York, NY, USA (2015)
4. Andriotis, P., Sasse, M., Stringhini, G.: Permissions snapshots: Assessing users' adaptation to the android runtime permission model. In: 2016 IEEE International Workshop on Information Forensics and Security (WIFS). p. To appear (Dec 2016)
5. Andriotis, P., Takasu, A., Tryfonas, T.: Smartphone message sentiment analysis. In: IFIP International Conference on Digital Forensics. pp. 253–265. Springer (2014)
6. Andriotis, P., Tryfonas, T.: Impact of User Data Privacy Management Controls on Mobile Device Investigations, pp. 89–105. Springer International Publishing, Cham (2016)
7. Beresford, A.R., Rice, A., Skehin, N., Sohan, R.: Mockdroid: Trading privacy for application functionality on smartphones. In: Proceedings of the 12th Workshop on Mobile Computing Systems and Applications. pp. 49–54. HotMobile '11, ACM, New York, NY, USA (2011)
8. Bettini, C., Riboni, D.: Privacy protection in pervasive systems: State of the art and technical challenges. Pervasive and Mobile Computing 17, Part B, 159 – 174 (2015), 10 years of Pervasive Computing' In Honor of Chatschik Bisdikian
9. Bugiel, S., Heuser, S., Sadeghi, A.R.: Flexible and fine-grained mandatory access control on android for diverse security and privacy policies. In: Presented as part of the 22nd USENIX Security Symposium (USENIX Security 13). pp. 131–146. USENIX, Washington, D.C. (2013)
10. Hornyack, P., Han, S., Jung, J., Schechter, S., Wetherall, D.: These aren't the droids you're looking for: Retrofitting android to protect data from imperious applications. In: Proceedings of the 18th ACM Conference on Computer and Communications Security. pp. 639–652. CCS '11, ACM, New York, NY, USA (2011)
11. Jung, J., Han, S., Wetherall, D.: Short paper: Enhancing mobile application permissions with runtime feedback and constraints. In: Proceedings of the Second ACM Workshop on Security and Privacy in Smartphones and Mobile Devices. pp. 45–50. SPSM '12, ACM, New York, NY, USA (2012)
12. Ongtang, M., McLaughlin, S., Enck, W., McDaniel, P.: Semantically rich application-centric security in android. Security and Communication Networks 5(6), 658–673 (2012)
13. Pearce, P., Felt, A.P., Nunez, G., Wagner, D.: Addroid: Privilege separation for applications and advertisers in android. In: Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security. pp. 71–72. ASIACCS '12, ACM (2012)
14. Wijesekera, P., Baokar, A., Hosseini, A., Egelman, S., Wagner, D., Beznosov, K.: Android permissions remystified: A field study on contextual integrity. In: 24th USENIX Security Symposium (USENIX Security 15). pp. 499–514. USENIX Association, Washington, D.C. (Aug 2015)