How do Apps Evolve in Their Permission Requests? A Preliminary Study

Paolo Calciati[♠,◇] Alessandra Gorla[♠]
{paolo.calciati,alessandra.gorla}@imdea.org
[♠]IMDEA Software Institute, Madrid, Spain [◇]Universidad Politécnica de Madrid, Spain

Abstract

In this preliminary study we try to understand how apps evolve in their permission requests across different releases. We analyze over 14K releases of 227 Android apps, and we see how permission requests change and how they are used. We find that apps tend to request more permissions in their evolution, and many of the newly requested permissions are initially overprivileged. Our qualitative analysis, however, shows that the results that popular tools report on overprivileged apps may be biased by incomplete information or by other factors. Finally, we observe that when apps no longer request a permission, it does not necessarily mean that the new release offers less in terms of functionalities.

1 Introduction

With auto-updates enabled by default on Android devices, it is easy for developers to produce and distribute new releases of their apps. Users can immediately use the new releases without even noticing the update process, unless there are changes in dangerous permission requests that require their approval. The most recent releases of the Android framework, although, no longer notify users about changes in the dangerous permissions list, unless the app requires a permission that belongs to a new permission group. For instance, an app declaring the RECEIVE_SMS permission, and thus capable of receiving SMS, could suddenly start sending SMS without further user approval by adding the SEND_SMS permission, since these two permissions belong to the same group. Changes in the permissions list are however worth notice, since they most likely represent a major change in the behavior of the app.

In this paper we present the work accepted for publication at the 14th International Conference on Mining Software Repositories [CG17], to be held in Buenos Aires in May 2017. We perform a preliminary study on the evolution of permission requests of Android apps. We analyze over 14K different releases of 227 apps, with a minimum of 50 releases per app, and we study how developers request legitimate permissions (i.e., requested and used in the code) and overprivileged permissions (i.e., declared in the manifest file but not used by the app).

This is not the first paper aiming to study the evolution of permission requests in Android apps. Tailor and Martinovic [TM16] studied over 1.6M apps, taking quarterly snapshots of the Google Play store for almost two years and analyzing the evolution of permission, with a focus on the *dangerous* category. Their analysis highlights trends in the store in terms of permission requests, such as free and popular apps being more likely to add new permissions, but does not look at single applications specifically. Krutz et al. [KMM+15] used different static analysis tools to analyze 4416 releases belonging to 1179 apps, creating a dataset containing information about applications' development and maintenance. Wei et al. [WGNF12] conducted a study similar to the one we present in this paper, but used a much smaller dataset with an average of 4 releases per app. They analyzed patterns and permission distributions, and reported that applications tend to be overprivileged and to request more permission over time. Moreover, they focused their interest in comparing pre-installed apps and third-party apps, while we are mainly interested in the latter.

With respect to the related work, our analysis fo-

Proceedings of the Seminar Series on Advanced Techniques and Tools for Software Evolution SATToSE 2017 (sattose.org).
07-09 June 2017, Madrid, Spain.

cused on studying a small set of apps but with a high number of releases, rather than simply taking in consideration the biggest possible APK set. To the best of our knowledge, ours is the first study to analyze such a big number of releases for single applications. Moreover, we present findings and qualitative analyses that other works do not cover.

Our empirical study highlights several interesting findings: we see a common trend of apps requiring more permissions over time, confirming what [WGNF12, TM16] observed; some dangerous permissions that are added in following releases do not need further approval from the user; there is a mild correlation between changes in target SDK and permission requests; a qualitative analysis of the results highlights that permission removal does not imply the loss of a functionality; last but not least, we identify several problems with Androguard, the state of the art tool to report overprivileged apps, showing that using it blindly can lead to biased results.

2 Dataset and Evaluation Process

To run our preliminary empirical evaluation, we resorted to the Androzoo dataset [ABKLT16], which comprises a collection of over 5 million Android apps retrieved from different sources. Since this study does not focus on malware, we selected the Google Play store as the only source to consider, given that several studies have shown that the apps distributed through this channel are largely trustworthy [ABJ+16, NZJ+14, ZWZJ12]. From this selection, we considered only apps with at least 50 releases, to have enough data for a study on apps evolution.

We analyzed each release with Androguard, a tool that can extract declared permissions and report the overprivileged ones[1]. In this study we focused only on Android official permissions, leaving out app-specific permissions. Androguard crashed while analyzing the Dalvik bytecode of several releases of 10 different applications. We decided to keep in our analysis only the apps for which Androguard managed to analyze at least 50 releases: with this additional pruning, our final dataset comprises 227 applications with a total of 14,450 releases. The distribution of releases per app goes from a minimum of 50 up to 171, with a mean value of 64. The APKs in the dataset have been released within the time-frame of August 2008 – January 2017.

To further understand the nature of the apps in our dataset, we looked at the category, number of downloads, and rating as reported in the Google Play store. The dataset is quite varied, since it covers all the 32 Android categories, but it focuses mainly on high quality and popular applications, with an average of over 44M downloads per app, and an average star rating of 4.29 out of 5. All applications we considered are still active and available on the Google Play store, except for 8, which either changed package name or have been discontinued.

[1] `https://github.com/androguard/androguard`

3 Empirical Results on Permissions Evolution

We first focus on a quantitative study to evaluate how permission requests evolve. We later conduct a qualitative study to have a better understanding of the phenomena observed in the first study.

3.1 Quantitative Study

The quantitative study aims to look at how permission requests change across the lifetime of an APK. We posed ourselves three research questions, which we are presenting in the following sections.

RQ1: How do permission requests evolve?

We first look at how the permissions list changes between a release and its following one. We observe that for 13,637 out of 14,223 release transitions, which accounts to 95.88% of the total number, there are no changes in the list of requested permissions. Out of the remaining 586 releases, we observe on average 0.64 permission additions per release, showing a clear trend of apps increasing the number of asked permissions over time. Most changes involve only adding a single permission (287 occurrences); we observe 78 occurrences of adding two permissions at the same time, and only 42 involving three or more permission added. Permission are rarely removed, and most of the times that involves a single permission.

Similarly to what Wei et al. did in their work [WGNF12], we look for specific patterns in our dataset and we find a similar distribution of the permission evolution patterns. In our analysis we also distinguish between orverprivileged and legitimate permissions: surprisingly, the most frequent pattern in our dataset is to add a permission that remains overprivileged for the whole lifetime of the app. The second most common pattern is to add a legitimate permission, and the third to have an overprivileged permission become legitimate. We further analyze these trends in Section 3.2.

RQ2: Which are the most frequently added permissions?

RQ1 showed that the most common trend is to add permissions, whether legitimate or overprivileged: in

Table 1 we look at which ones are most frequently added among the 646 permission addition found.

Table 1: Most Frequently added permission

Permission (legitimate)	Usage
ACCESS_COARSE_LOCATION	58 (19.33%)
READ_PHONE_STATE	49 (16.33%)
ACCESS_FINE_LOCATION	49 (16.33%)
CAMERA	32 (10.67%)
READ_CONTACTS	31 (10.33%)

Permission (overprivileged)	Usage
READ_EXTERNAL_STORAGE	112 (32.37%)
WRITE_EXTERNAL_STORAGE	25 (7.23%)
READ_SMS	25 (7.23%)
CAMERA	25 (7.23%)
RECEIVE_SMS	24 (6.94%)

Taking into account the recent changes in Android 6 explained in Section 1, we further analyze each newly added permission, and verify if it belongs to a permission group already granted in the previous release: in 35.75% of the cases the newly added permission indeed falls in the same group of one of the already granted permission. READ_EXTERNAL_STORAGE is the permission for which this behavior happened the most: many apps add this permission while they already declare WRITE_EXTERNAL_STORAGE. What is apparently unclear to developers is that apps granted with WRITE_EXTERNAL_STORAGE implicitly have read access as well. A more interesting case regards accessing the user's location: many applications initially just require ACCESS_COARSE_LOCATION permission, which only gives an approximate location, and add the ACCESS_FINE_LOCATION afterwards, thus obtaining the precise GPS location without any user interaction.

RQ3: Do permission changes concentrate in specific time periods?

The last step of our quantitative analysis looks at the distribution of releases over time: we want to see if releases are more frequent in specific time frames. We thus divide the time frame in year quarters (x axis in Figure 1), and we report the percentage of permission changes that we observe in that quarter (blue bars in the plot).

We can observe that there are peaks in the number of releases in q3 2013 and q3 2015, in our opinion due to the low amount of releases in the preceding quarters. To further understand this trend, we list changes occurring in two subsequent releases that involve either the target or the minimum Android SDK. We plot, as yellow bars, the percentage distribution of

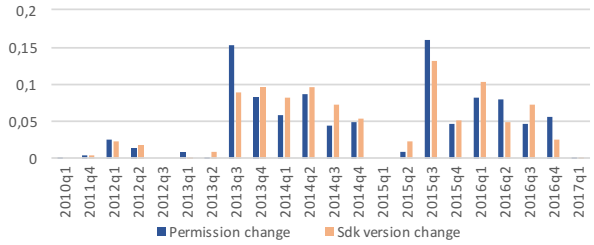

Figure 1: Permission change vs. Android SDK version change.

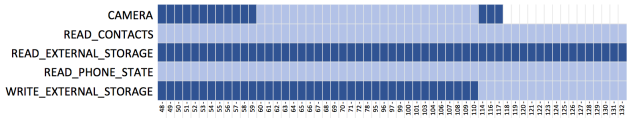

Figure 2: Permission evolution plot of the TabShop app.

such changes on the same time frame in Figure 1. We see that the two distributions have similar trends, suggesting that a change in the minimum or target SDK might trigger permission changes. We checked whether high change picks would correspond to release dates of new Android SDKs, but we could not find a positive correlation.

3.2 Qualitative Analysis

The results of the quantitative analysis reported in Section 3.1 show that several apps have interesting patterns in their permission requests. Some apps, for instance, add and remove the same permission multiple times in their lifetime, as if some functionalities were added and removed from the app. In many other cases, instead, developers seem to ask for permissions they never use. To further analyze the permission evolution of the apps in our dataset, we plot legitimate and overprivileged permission requests as in Figure 2, reporting on the y-axis the permissions that the app requires, and on the x-axis all the releases of the app. We use white, light blue and dark blue to represent not requested, legitimate and overprivileged permissions, respectively.

Figure 2 is the permission evolution of TabShop, a free shop keeping, cashier and point of sale (POS) app. Among the functionalities it offers, it can scan barcodes. CAMERA thus seems an essential permission for the whole lifetime of this app. The plot, instead, reports that the CAMERA permission is initially requested but not used, it is later legitimately used for several releases, and finally it disappears after a few more releases that see it overprivileged. We installed and ran different APKs of this app in the emulator, and found out that in the first releases it uses an intent to launch "Barcode Scanner", an alternative app

to scan barcodes. Between release 60 and 110 the app instead implements the feature itself, thus using the device's camera directly, however Within this timeframe the CAMERA permission appears twice in the manifest. From version 114 to version 117 the app switches back to the "Barcode Scanner" intent, but removes only one CAMERA permission from manifest, thus leaving the duplicate and making it overprivileged. Developers finally removed the unnecessary permission in version 118. The qualitative analysis on this app shows us that if an app removes a permission it does not necessarily mean that it no longer offers the same functionality, but rather may implement it in an alternative way. This app is not the only one in our dataset that declares *duplicate permissions*: More precisely, 52 apps of the 227 we analyzed show this problem in some of their releases, with Firefox Beta topping the list with duplicated permissions in over 100 releases. Similarly, we observe that many apps include misspelled permissions (e.g. VIBRATION instead of VIBRATE). Although these cases are harmless, since the app asks for a permission that either does not exist or it already uses, they represent a bad practice, and may falsely report alarms.

Furthermore, we notice that Androguard incorrectly reports other permissions as overprivileged. This happens for example for WRITE_EXTERNAL_STORAGE, which is also one of the most frequently reported permissions as overprivileged. We install the app com.popularapp.periodcalendar, v9, released in 2013-02 on an emulator, removing the permissions, labeled as overprivileged. Running the repackaged app leads to a crash when it tries access the SD card, showing that the permission is legitimate. The incorrect information reported by Androguard may be due to different causes: 1) it is known that a complete and correct mapping between API calls and permissions does not exist yet [AZHL12, FCH+11, BKMLT12]; 2) I/O operations may be implemented in native code, preventing Androguard from seeing them; 3) finally, we identified a bug in Androguard, which causes the analysis to use the API mappings relative to API 19, even when the app targets another release. Thus, the lesson we learned is that trusting tools such as Androguard blindly may bias significantly the results.

4 Conclusions

We presented an overview on the evolution of Android permission over a dataset of application with a large number of versions available. In our quantitative analysis we see a common trend of apps requiring more permissions over time, confirming similar studies in the literature. We found new evolution patterns, and a mild correlation between changes in target SDK and in permission requests. The qualitative analysis highlighted some confusion between developers regarding the use of permissions, as suggested by our finding of mislabeled, duplicated and overprivileged permissions. Finally, we discovered that the removal of a permission does not imply the loss of a functionality. Last but not least, we identified several problems with Androguard, the state of the art tool to report overprivileged apps, showing that using it blindly can lead to biased results.

Acknowledgements

This work was supported by the EU FP7-PEOPLE-COFUND project AMAROUT II (n. 291803), by the Spanish project DEDETIS, and by the Madrid Regional project N-Greens Software (n. S2013/ICE-2731).

References

[ABJ+16] Kevin Allix, Tegawendé F. Bissyandé, Quentin Jérome, Jacques Klein, Radu State, and Yves Le Traon. Empirical assessment of machine learning-based malware detectors for android - measuring the gap between in-the-lab and in-the-wild validation scenarios. *Empirical Software Engineering*, 21(1):183–211, 2016.

[ABKLT16] Kevin Allix, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. AndroZoo: Collecting millions of android apps for the research community. In *Proceedings of the 13th International Conference on Mining Software Repositories*, MSR '16, pages 468–471, New York, NY, USA, 2016. ACM.

[AZHL12] Kathy Wain Yee Au, Yi Fan Zhou, Zhen Huang, and David Lie. PScout: analyzing the Android permission specification. In *Proceedings of the 19th Conference on Computer and Communications Security (CCS)*, pages 217–228, New York, NY, USA, 2012. ACM.

[BKMLT12] Alexandre Bartel, Jacques Klein, Martin Monperrus, and Yves Le Traon. Automatically securing permission-based software by reducing the attack surface: An application to Android. In *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 274–277, 2012.

[CG17] Paolo Calciati and Alessandra Gorla. How do apps evolve in their permission

requests? a preliminary study. In *Proceedings of the 14th International Conference on Mining Software Repositories*, MSR '17, 2017.

[FCH+11] Adrienne Porter Felt, Erika Chin, Steve Hanna, Dawn Song, and David Wagner. Android permissions demystified. In *Proceedings of the 18th Conference on Computer and Communications Security (CCS)*, pages 627–638, New York, NY, USA, 2011. ACM.

[KMM+15] Daniel E. Krutz, Mehdi Mirakhorli, Samuel A. Malachowsky, Andres Ruiz, Jacob Peterson, Andrew Filipski, and Jared Smith. A dataset of open-source android applications. In *Proceedings of the 12th Working Conference on Mining Software Repositories*, MSR '15, pages 522–525, Piscataway, NJ, USA, 2015. IEEE Press.

[NZJ+14] Yi Ying Ng, Hucheng Zhou, Zhiyuan Ji, Huan Luo, and Yuan Dong. Which android app store can be trusted in china? In *Proceedings of the 2014 IEEE 38th Annual Computer Software and Applications Conference*, COMPSAC '14, pages 509–518, Washington, DC, USA, 2014. IEEE Computer Society.

[TM16] Vincent F. Taylor and Ivan Martinovic. A longitudinal study of app permission usage across the google play store. *CoRR*, abs/1606.01708, 2016.

[WGNF12] Xuetao Wei, Lorenzo Gomez, Iulian Neamtiu, and Michalis Faloutsos. Permission evolution in the android ecosystem. In *Proceedings of the 28th Annual Computer Security Applications Conference*, ACSAC '12, pages 31–40, New York, NY, USA, 2012. ACM.

[ZWZJ12] Yajin Zhou, Zhi Wang, Wu Zhou, and Xuxian Jiang. Hey, you, get off of my market: Detecting malicious apps in official and alternative android markets. In *NDSS*. The Internet Society, 2012.