Who Added that Permission to My App? An Analysis of Developer Permission Changes in Open Source Android Apps

Daniel E. Krutz, Nuthan Munaiah, Anthony Peruma, and Mohamed Wiem Mkaouer
Department of Software Engineering
Rochester Institute of Technology
Rochester, NY, USA
Email: {dxkvse, nm6061, axp6201, mwmvse}@rit.edu

***Abstract*—Android applications rely on a permission-based model to carry out core functionality. Appropriate permission usage is imperative for ensuring device security and protecting the user's desired privacy levels. But who is making the important decisions of which permissions the app should request? Are they experienced developers with the appropriate project knowledge to make such important decisions, or are these crucial choices being made by those with relatively minor amounts of contributions to the project? When are these permission-related decisions being made in the app's development life cycle? We examined 1,402 Android version control repositories containing over 331,318 commits including 18,751 AndroidManifest.xml versions to better understand when, why, and who is adding permissions to apps. We found that (I) developers with more experience are more likely to make permission-based changes (II) permissions are typically added earlier in apps' commit lifetime, but their removal is more sustained throughout the commit lifetime (III) developers reverting permission-based changes are typically more experienced than developers who initially made the change being reverted.**

I. Introduction

Android employs a permission-based system where apps require specific permissions to carry out specific functionalities. Developers must explicitly state the permissions an app may request, while the end user must accept any requested *dangerous* permissions. Some permissions include the ability to read SMS messages, record audio through the phone's microphone, and access the user's location [5].

The decision of which permissions an app should have access to should not be taken lightly since they may disrupt the user and carry a variety of possible security and functional implications. Some of which include under and over-permissions, increased app susceptibility to malware and unwanted data leakage to ad libraries [11]–[13], [25]. But who is making these important decisions about the permissions an app should request and at what points in the software development process are these decisions being made? *The goal of our work is to better understand how permissions fit into the development process.* In this work, we discuss our analysis of more than 1,400 open source Android app repositories. Our work is guided by the following research questions:

RQ0 Permission Exploration: Who tends to make permission-based changes to apps and when are such changes typically made in apps' commit lifetime?
RQ1 Permission Reversion Frequency: How often are permission-based changes to apps reverted and who is making these decisions?

> *Our primary findings suggest that a wide range of developers typically make permission-based changes, but developers with higher project ownership scores were more likely to revert existing permissions.*

The rest of the paper is organized as follows: In Section II we discuss related work and Section III provides background information on Android permissions. Section IV presents our collection and analysis process, while Section V addresses our research questions. Section VI discusses limitations and future work to be conducted, and Section VII concludes our work.

II. Related Work

There has been a substantial amount of work in understanding why Android permissions are inappropriately used, and the negative implications misuse carries. Grace et al. [13] conducted work on permissions probing, which is when a 3rd party component attempts to use a permission in the hope that the attached app has requested it from the user. If the attached app has requested a permission, then the component will also have access to that permission as well. This is often done to collect, and transmit potentially sensitive information which should not be normally available to the 3rd party component. This work found that more than half of all ad libraries try to probe for open permissions.

Stevens *et al.* [22] analyzed 10,000 free Android apps and found a strong sub-linear relationship between the popularity of a permission and the frequency of its misuse. They found that developers were more likely to misuse a permission when they did not understand it, and that the popularity of a permission is strongly associated with its misuse. A powerful method of avoiding permission misuse is through developer education and community support. Krutz *et al.* [14] created a

978-1-5386-2669-6/17 $31.00 © 2017 IEEE
DOI 10.1109/MOBILESoft.2017.5

public dataset of over 1,100 Android apps from the F-Droid [2] repository, but only analyzed the apps using existing static analysis tools.

Many projects have analyzed software repositories to examine questions ranging from if the commit message of an app had a correlation with the quality of the commit [19] to analyzing developer sentiment in commit logs [21]. However, our work indicates the first known of its kind to examine permissions using the version control history of a project. Although previous works have examined developer code ownership and their implications [15], [17], [18], none have focused on mobile applications as was done in our study.

III. Android Permissions

Android apps require specific permissions to carry out specific functionality. An objective of this system is the the *principle of least privilege* or granting an app the least amount of privilege that it needs to properly function [11]. This is intended to not only limit the access an app has to unintended permissions, but limit the effects that malware may have on a device [10], [20]. For example, in order for an app to read SMS messages, it must request the `READ_SMS` permission. The *AndroidManifest.xml* file contains all requested permissions for an app. While many permissions are considered to be less risky, others carry significantly more potentially hazardous risks and are known as *Dangerous* permissions [5].

Deciding on the permissions an app should request is considered to be one of the most sensitive activities undertaken during app development due to the potential security risks [11], [16] and possible negative effects on the user's perception of the app [8]. Recent studies revealed that developers frequently misuse permissions by either not adding enough permissions to support requested functionality, or by adding unnecessary permissions that are not needed by any components in the app. This problem may be caused by a variety of factors including a lack of permissions based knowledge by the developers [22]. Unfortunately, there is no permission enforcement mechanisms in Google Play, which frequently gives developers too much freedom when posting apps to the Google Play store [7].

IV. Data Collection and Analysis

Our first step was to collect open source Android repositories from F-Droid [2], a popular open source repository of Android apps. We collected the git repositories for each app, which contains all version control information about each project and range from 2009 to present in numerous app categories. When collecting permissions, we recorded all permissions, including those which were custom. Table I shows an overview of collected data.

At the time of our analysis, *F-Droid* contained information for 2,372 open source Android apps. From this resultset our tool scanned for the existence of the *AndroidManifest.xml* file along with the commit history of the file. This process identified 1,402 apps that had a *AndroidManifest.xml* file with a history of commits. We were unable to collect the repositories of the other projects since they were not publicly accessible, or located in GitHub.

TABLE I: Data Overview

Item	Count
Analyzed Apps	1,402
Unique Permissions (including custom)	238
Manifest File Versions	18,751
Number of Project Committers	11,920
Permissions Added	4,494
Permissions Removed	480
Total Project Commits	331,318

To analyze the collected repositories, we created a tool known as Open Source Android Repository Analyzer (oSARA) [4]. Using this tool, we extracted version control commit information such as when the commit was made and by whom. The tool then extracts all committed *AndroidManifest.xml* files from the version control history and records all modified permissions in these files. Using this collected information, *oSARA* then determines all altered permissions, who made the alterations, and when they were made. The committed version of the *AndroidManifest.xml* file was also extracted from the repositories, and all metadata was stored in a SQLite database. The tool ran for over 24 hours on a dedicated server to download approximately 50 GB of repository content and then analyze its commit history. Using this information, we were able to correlate the altered Android permissions with a specific commit in the version control repository. We then calculated the *Developer's Commit Ratio (DCR)* for each app, which is defined as: DCR = $(\frac{IndividualcontributorCommits}{TotalAppCommits})$. This ratio represents the number of contributions made by a given developer for a project divided by the number of all commits done by all project's contributors. In other terms, it represents the contribution share of every author for a given project. In this work, we use DCR as a proxy for developer experience within a project. For each committed *AndroidManifest.xml* file that contained altered permissions, we recorded this DCR value. We only examined the master branch for each git repository, and only considered apps with at least 2 committers.

To provide researchers with more data points to develop new hypotheses, metadata of the apps were obtained from the Google Play store. Presently, Google does not offer an API to retrieve this data, so a web scraping tool was built to scrape the metadata from the app's Google Play web page. The use of text files by F-Droid to hold app related metadata proved to be challenging as multiple regular expressions had to be written to parse the files.

V. Evaluation

We next addressed our research questions using this data:

RQ0 Permission Exploration: *Who tends to make permission-based changes to apps and when are such changes typically made in an apps' commit lifetime?*

For RQ0, we explored certain aspects (such as who and when) of permission-based changes to apps. To understand the developers making permission-based change to apps, we compared the distributions of DCR for developers who had made permission-based changes to apps and those who had not. Since we are comparing the DCR of developers across all the apps in our data set, the number of developers who contribute to each application is a confounding factor. We accounted for this by dividing the DCR of a developer by the number of developers who had contributed to the app.

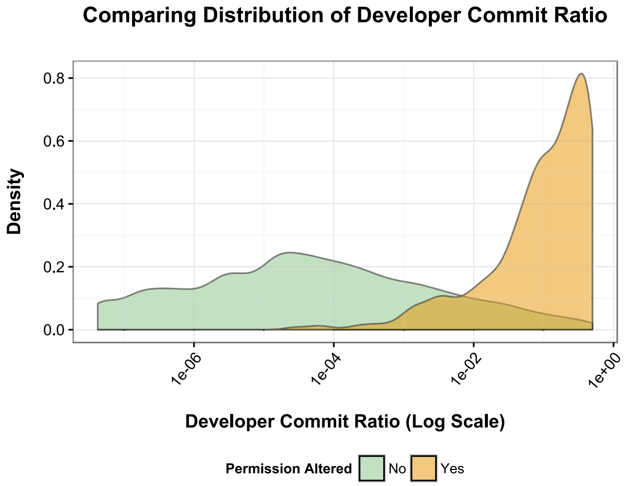

Fig. 1: Comparing the distribution of DCR (controlled for number of developers contributing to an app) of developers who had altered permission(s) and developers who had not

Figure 1 shows a plot comparing the two distributions and shows that developers who made permission-based changes have a higher DCR than developers who did not. To quantitatively assess the difference in distribution, we used the non-parametric Mann-Whitney-Wilcoxon (MWW) test to assess if there was an association between DCR and the likelihood of a developer making permission-based changes. The outcome from the MWW test revealed, not surprisingly, that developers with a higher DCR were more likely to make permission-based changes to apps than developers with lower DCR. The outcome was statistically significant with p-value $\ll 0.01$. We went further to analyze if there was any difference between the DCR of developers who added permissions and that of developers who removed permissions. The outcome from the MWW test was not statistically significant indicating that the distributions of DCR of developers who added permissions is similar to that of developers who removed permissions.

To understand when, in apps' commit lifetime, are permission-based changes made to apps, we computed the time when each permission-based change was made to an app. The time of a permission-based change was computed as the number of days from the first commit to the apps' source code. Here the commit lifetime of an app is a confounding factor, therefore, we expressed the time of a permission-based change relative to the apps' commit lifetime (i.e. the number of days between the first and last commit to the apps' source code). To understand the chronology of permission-based changes at a lower level of granularity, we grouped the changes by permission type and segregated the changes into additions and removals. Figure 2 shows the distribution of the times of permissions change. As seen in the figure, the addition of permissions, regardless of the type, are typically done when an app is fairly new and further additions tend to be spread over the entirety of the commit lifetime of the app. The removal of permissions, on the other hand, is observed throughout apps' commit lifetime.

RQ1 Permission Reversion Frequency: *How often are permission-based changes to apps reverted and who is making these decisions?*

In RQ0, we found that there is no statistically significant difference between the DCR of developers adding a permission and that of developers removing a permission. In RQ1, we introduced a temporal factor into the analysis. In other words, we wanted to investigate the frequency with which permission-based changes are reverted i.e. a permission added by a developer is later removed by another developer, or vice versa. We further wanted to compare the DCR of the developer reverting the permission-based change to that of the developer who initially made the change.

TABLE II: Reversion frequency of permission-based changes

Reversion Type	# Occurrences	# Higher DCR
Removal	106	61 (57.54%)
Addition	33	22 (66.67%)

Table II displays the number of occurrences when permission-based changes were reverted. In more than half the reversions (# Higher DCR) the DCR of the developer reverting the permission-based change is higher than that of the developer who initially made the change. This shows that developers with lower commitments to the project tend to make permission-related decisions that were later reverted by more experienced developers. This interesting observation needs to be further investigated to decipher the motivation behind introducing the permission change and also the reasoning behind its revert. The average difference in DCR between the reverting developer and the developer who initially made the change was 0.4184 and 0.4113 for removal and addition. The top three permissions later removed by developers with higher DCR were `READ_EXTERNAL_STORAGE`,`RECEIVE_BOOT_COMPLETED`, and `WRITE_EXTERNAL_STORAGE`. The top three permissions later added by developers with higher DCR were `ACCESS_NETWORK_STATE`, `ACCESS_MOCK_LOCATION` and `WAKE_LOCK`. Further analysis is needed to explain the probable reasons for a permission-based change of a developer with lower DCR being reverted by one with a higher DCR. One possible reason could be that the initial permission change may have been done in error (an under or over-privilege) with the reversion being the corrective action. A more in-depth analysis of the commit message

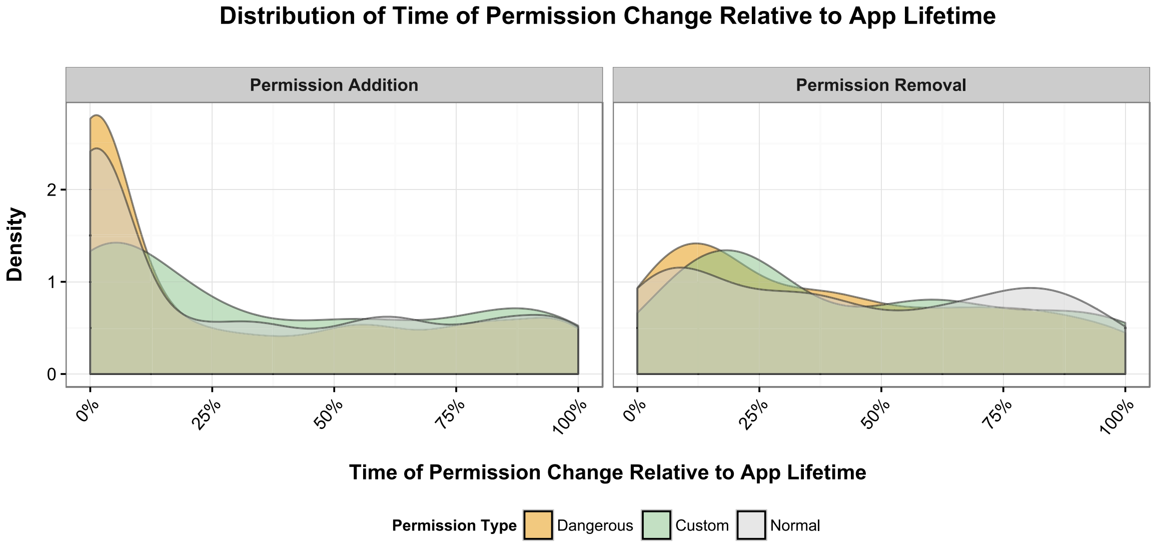

Fig. 2: Distribution of times of permissions change relative to apps' commit lifetime for addition and removal of permissions

when the permission-based change was reverted may be first step to understanding the phenomenon. A cursory analysis of some of the commit messages did not reveal the possibility of erroneous commits but the messages were too brief to conclusively say one way or the other.

VI. Limitations and Future Work

Although our results provided several interesting findings, there are areas which can be more thoroughly explored and elaborated upon. In our analysis, we studied open source Android apps in a variety of categories, but only from a single source (F-Droid). In future work, we will expand our app selection and include apps collected from other sources as well. However, this could be difficult since finding a substantial number of other open source repositories for Android apps may be a challenging process.

Like most research which examines version control systems, we are unable to definitively determine who is actually making each commit. The same committer could use different accounts, or developers could be using pair-programming, thus creating a situation where two developers modify the software, but only one is given credit for the submission.

Previous work has analyzed apps by examining version control repositories and many studies have integrated developer interviews or user surveys in their analysis [23], [24]. Our work should be expanded to include more developer interviews and more qualitative information. Further work could be done to examine apps at different development stages using various static analysis tools to examine a variety of security and quality based metrics of each app. Some potential tools include PScout [6], FindBugs [3], AndroGuard [1], StaDynA [26], or TaintDroid [9]. Android apps often suffer from permission-misuse where apps request too few, or too many permissions [11], [13]. Future work may be done to analyze if the DCR score of the author of a permission change correlates to a higher rate of permission misuse. This analysis may be conducted using a permissions analysis tool such as PScout [6]. Future work could determine if more mature apps, ones who have had longer version histories, tend to have high rates of permission misuse in comparison to new apps. Research may also be conducted to determine at what phases under and over-privileges are being added to apps.

VII. Conclusion

Summary: We examined the version control repositories of 1,402 Android apps to better understand how permissions decisions are being made. We created a publicly accessible tool *oSARA* to analyze the repositories.

Findings: In the exploration question, we found that the developers with higher DCR were more likely to make permissions-based changes to apps than other developers. Among the developers who made permissions-based changed, we found no difference in the DCR of developers who add permissions and those who remove permissions. Analyzing the chronology of the permission-based changes, we found that the addition of permissions is typically done earlier in apps' commit lifetime. However, the removal of permissions is a more sustained activity throughout apps' commit lifetime.

In the reversion question, when we introduced a temporal factor in analyzing the reversion of permissions-based changes, we found that in more than half of the permissions-based changes in which a developer reverted a permission-based change, the DCR of the developer making the reversion was higher than the developer who initially made the permissions change that was reverted.

References

[1] Androguard. https://code.google.com/p/androguard/.

[2] F-droid - free and open source android app repository. https://f-droid.org.

[3] Findbugs. http://findbugs.sourceforge.net/.

[4] osara: Open source android repository analyzer. https://github.com/dan7800/oSARA.

[5] System permissions. https://developer.android.com/guide/topics/security/permissions.html.

[6] K. W. Y. Au, Y. F. Zhou, Z. Huang, and D. Lie. Pscout: Analyzing the android permission specification. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 217–228, New York, NY, USA, 2012. ACM.

[7] D. Barrera, J. Clark, D. McCarney, and P. C. van Oorschot. Understanding and improving app installation security mechanisms through empirical analysis of android. In *Proceedings of the Second ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, SPSM '12, pages 81–92, New York, NY, USA, 2012. ACM.

[8] S. Egelman, A. P. Felt, and D. Wagner. Choice architecture and smartphone privacy: There's a price for that. In *In Workshop on the Economics of Information Security (WEIS)*, 2012.

[9] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth. Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones. In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, OSDI'10, pages 393–407, Berkeley, CA, USA, 2010. USENIX Association.

[10] P. Faruki, A. Bharmal, V. Laxmi, V. Ganmoor, M. S. Gaur, M. Conti, and M. Rajarajan. Android security: a survey of issues, malware penetration, and defenses. *IEEE communications surveys & tutorials*, 17(2):998–1022, 2015.

[11] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner. Android permissions demystified. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, pages 627–638, New York, NY, USA, 2011. ACM.

[12] X. Gao, D. Liu, H. Wang, and K. Sun. Pmdroid: Permission supervision for android advertising. In *2015 IEEE 34th Symposium on Reliable Distributed Systems (SRDS)*, pages 120–129, Sept 2015.

[13] M. C. Grace, W. Zhou, X. Jiang, and A.-R. Sadeghi. Unsafe exposure analysis of mobile in-app advertisements. In *Proceedings of the Fifth ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WISEC '12, pages 101–112, New York, NY, USA, 2012. ACM.

[14] D. E. Krutz, M. Mirakhorli, M. S. A., A. Ruiz, J. Peterson, A. Filipski, and J. Smith. A dataset of open-source android applications. In *Proceedings of the 12th Working Conference on Mining Software Repositories*. ACM, 2015.

[15] M. Linares-Vsquez, K. Hossen, H. Dang, H. Kagdi, M. Gethers, and D. Poshyvanyk. Triaging incoming change requests: Bug or commit history, or code authorship? In *2012 28th IEEE International Conference on Software Maintenance (ICSM)*, pages 451–460, Sept 2012.

[16] P. Manadhata and J. Wing. An attack surface metric. *Software Engineering, IEEE Transactions on*, 37(3):371–386, May 2011.

[17] M. E. Nordberg. Managing code ownership. *IEEE software*, 20(2):26–33, 2003.

[18] F. Rahman and P. Devanbu. Ownership, experience and defects: A fine-grained study of authorship. In *Proceedings of the 33rd International Conference on Software Engineering*, ICSE '11, pages 491–500, New York, NY, USA, 2011. ACM.

[19] E. A. Santos and A. Hindle. Judging a commit by its cover: Correlating commit message entropy with build status on travis-ci. In *Proceedings of the 13th International Conference on Mining Software Repositories*, MSR '16, pages 504–507, New York, NY, USA, 2016. ACM.

[20] J. Sellwood and J. Crampton. Sleeping android: The danger of dormant permissions. In *Proceedings of the Third ACM Workshop on Security and Privacy in Smartphones &#38; Mobile Devices*, SPSM '13, pages 55–66, New York, NY, USA, 2013. ACM.

[21] V. Sinha, A. Lazar, and B. Sharif. Analyzing developer sentiment in commit logs. In *Proceedings of the 13th International Conference on Mining Software Repositories*, MSR '16, pages 520–523, New York, NY, USA, 2016. ACM.

[22] R. Stevens, J. Ganz, V. Filkov, P. Devanbu, and H. Chen. Asking for (and about) permissions used by android apps. In *Mining Software Repositories (MSR), 2013 10th IEEE Working Conference on*, pages 31–40, 2013.

[23] J. Tsay, L. Dabbish, and J. Herbsleb. Let's talk about it: Evaluating contributions through discussion in github. In *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2014, pages 144–154, New York, NY, USA, 2014. ACM.

[24] B. Vasilescu, K. Blincoe, Q. Xuan, C. Casalnuovo, D. Damian, P. Devanbu, and V. Filkov. The sky is not the limit: Multitasking across github projects. In *Proceedings of the 38th International Conference on Software Engineering*, ICSE '16, pages 994–1005, New York, NY, USA, 2016. ACM.

[25] M. Wei, X. Gong, and W. Wang. Claim what you need: A text-mining approach on android permission request authorization. In *Global Communications Conference (GLOBECOM), 2015 IEEE*, pages 1–6. IEEE, 2015.

[26] Y. Zhauniarovich, M. Ahmad, O. Gadyatskaya, B. Crispo, and F. Massacci. Stadyna: Addressing the problem of dynamic code updates in the security analysis of android applications. In *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, CODASPY '15, pages 37–48, New York, NY, USA, 2015. ACM.