Most Recent Version is in Overleaf: Who Added that Permission to My Android App? An Analysis of Developer Permission Changes in Open Source Android Apps.

XXXXXXXXXXXXXXX
XXXXXXXXX
XXXXXXXXX
XXXXXXXXX, XX, XXX
XXXXXX@XXXXX.XXX

ABSTRACT

Android applications (apps) rely on a permission-based model to carry out core functionality. Appropriate permission usage is imperative for ensuring proper 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 will only minor amounts of project experience? When are these permission decisions being made in the app's development lifecycle? Are they added at the beginning of the project, or are they steadily added throughout the lifecycle of the app?

In this work, we examine a large set of open source Android version control repositories to better understand when, why, and who is adding permissions. Our primary findings include: I) There is no significant correlation between the ratio of permissions-based commits and overall project commits by a developer. Our data indicates that developers with a low number of overall project commits make a disproportionately high number of permissions-related commits. II) There is no significant correlation between an app's user rating and the ratio of permissions-altering commits to overall commits. III) Permissions are added at a reasonably consistent rate throughout the software development process.

1. INTRODUCTION

Android is the world's most popular mobile OS [7] with over 1.8 million apps available from Google Play alone [2]. A cornerstone of Android security is its usage of permissions. Android apps do not have any default permissions associated with it. A developer must explicitly state the permissions an app must request, while the end user must accept any requested 'dangerous' permissions, some of which include the ability to read SMS messages, record audio through the phone's microphone, and access the user's location [3].

The decision of which permissions an app should have access to should not be taken lightly since they carry a variety of possible security and functional implications. Some of which include under & over-permissions, increased app susceptibility to malware and unwanted data leakage to ad libraries [10, 13]. But who is making these important decisions about the permissions an app should request? Who is deciding what gateways the app should request to make into our digital lives? Developers often rely upon user ratings of their software in the app stores in order to remain successful [4, 17]. Is there any correlation between what developer project experience levels are making permission-based decisions, and this all important user rating? To more properly address these permissions-based issues, we need to understand more about permissions in the app development process. In the following work, we describe our examination of open source Android app repositories in order to better understand how permissions fit into the development process. Our work is guided by the following research questions:

RQ1: *What is the correlation strength between the overall number of commits by a developer, and permission-based commits?* Our work indicates that developers with a low number of project commits make a disproportionately high number of permissions-based project decisions. This indicates that developers with small amounts of project experience are making a disproportionately high number of project based decisions.

RQ2: *Is there a correlation between the ratio of developer permission-based commits to overall project commits, and user ratings?* We found virtually no relationship between the ratio of developer permission-based commits to overall project commits, and user ratings. This indicates that there is no strong correlation between who is adding the permissions to an app, and its user rating.

RQ3: *At what development stages are permissions being added to Android projects?* We found that permissions are added at a fairly consistent level throughout the development life cycle of an app. This indicates that permission-based decisions are steadily made throughout the development of the app.

SAC'17, April 3-7, 2017, Marrakesh, Morocco

http://dx.doi.org/xx.xxxx/xxxxxxx.xxxxxxx

The rest of the paper is organized as follows: In Section 2 we discusses related work and Section 3 describes the permission structure of Android apps. Section 4 presents our collection and analysis process, while Section 5 addresses our research questions. Section 6 discusses our public data set and tool which we hope can be used to assist others in their research. Section 7 discusses future work to be conducted in this area, while Section 8 concludes our work.

2. 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. They found that more than half of all ad libraries try to probe for open permissions. This could often be the cause of an under-permission in an app since the ad library will try to use a permission which the developer did not request.

Stevens *et al.* [23] 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.* [19] created a public dataset of over 1,100 Android apps from the F-Droid[1] repository. This research focused more on the life cycle of the apps and how each iteration of the app evolved with every version control commit and placed more of an emphasis on analyzing the app from other quality perspectives.

Previous works have analyzed effects of permissions on the user's perception of the app. Lin et al. [20] examined user comfort levels when using permissions they did not fully understand, or when they did not comprehend why the app needed the permission. They found that users generally felt uncomfortable and may even delete applications when they did not understand why it requested a permission they deemed unnecessary. Egelman et al. [8] found that approximately 25% of users were typically willing to pay a premium in order to use the same application, but with fewer permissions, while about 80% of users would be willing to allow their apps more permissions to receive targeted advertisements if it would save them .99 cents on the purchase of the app. Contrary to these findings, other research has argued that users typically pay little attention to permissions when installing an app, and often do not understand or care about the precise functionality for most of the granted permissions [11]. Kelley et al. [16] conducted semi-structured interviews with Android users, and found that users paid limited attention to permission screens, and had poor understanding of what these permissions implied.

App ratings have demonstrated their importance in other areas of research as well. Harman et al. [15] found a strong correlation between the rating and the number of app downloads. Linares-Vasquez et al. [21] found that fault-proneness of the APIs used by the apps negatively impacts their user ratings. Khalid et al. [17] examined 10,000 apps using FindBugs and found that warnings such as 'Bad Practice', 'Internationalization', and 'Performance'categories are typically found in low-rated apps. They found that app developers could use static analysis tools, such as FindBugs, to repair issues before users complained about these problems. Even though we too use ratings as an evaluation measure, unlike earlier works we look at permission and security risks.

3. ANDROID PERMISSIONS

Android apps require specific permissions to carry out specific functionality. A primary goal of this system is the adherence to the *principle of least privilege* or granting an app the least amount of privilege that it needs to properly function. 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. For example, in order for an app to read SMS messages, it must request the `READ_SMS` permission, and to use the camera the app would request the `CAMERA` permission. The *AndroidManifest.xml* file contains all requested permissions for an app. While many permissions are considered to be less risky or *Normal* permissions, others carry significantly more potentially hazardous risks and are known as *Dangerous* permissions. Users must explicitly allow the app to make use of these more threatening permissions. 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 [10,22] and possible negative effects on the user's perception of the app [8]. In our analysis, we recorded both Normal and Dangerous permissions which were requested by the app.

Developers frequently make permissions based mistakes by adding too many, or too few permissions to an app. This may be caused by a variety of factors including a lack of permissions based on knowledge by the developers [23]. Unfortunately, there is no permissions enforcement mechanisms in Google Play, which frequently gives developers too much freedom when posting apps to the Google Play store [6]. In this study, we use the term *over-permission* to describe a permission setting that grants more than what a developer needs for the task. Likewise, an *under-permission* is when an app was not given all the permissions that it needs to properly function as intended.

4. DATA COLLECTION AND ANALYSIS

Following a process similar to one defined by Krutz et al. [19], our first step was to collect open source Android repositories from F-Droid, which hosts over 1,100 open source Android apps. We collected the complete git repositories for each app, which contains all version control information about each project. In order to analyze the collected reposi-

[1]https://f-droid.org/

tories, we created a tool known as 'the Open Source Android Repository Analyzer (oSARA). Using this tool, we extracted version control commit information such as when the commit was made and who the committer was. The committed version of the AndroidManifest.xml file was also extracted from from the repositories, and all metadata was stored in a SQLite database. 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 = $(\frac{IndividualAuthorCommits}{TotalAppCommits})$. For each committed AndroidManifest.xml file that contained altered permissions, we recorded this DCR value. In this analysis, we only examined the master branch for each git repository, and only considered apps with at least 2 committers and at least 10 commit updates to the AndroidManifest.xml file. This left us with a total of 481 apps and a total of 13,036 manifest commits.

We then determined the average Google Play user rating for each of the collected apps, which we found by matching the retrieved package and app name to the corresponding app in Google Play. We did not create associations when we did not find matches for both the package and app name, and excluded this information from our analysis. We also excluded apps with less than 25 user reviews, leading to a total of 96 apps. This sample size is larger or similar in size to other studies which have examined user feedback in Android apps [14, 18]. An overview of the data collection and analysis process is shown in Figure 1.

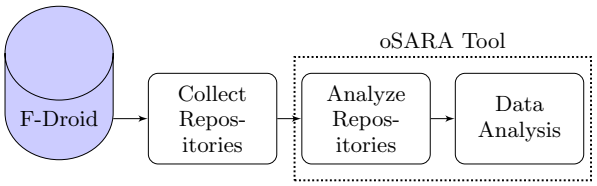

Figure 1: App Repository Collection and Analysis Process

5. EVALUATION

Using the collected data, we were then able to address our research questions.

RQ1: What is the correlation strength between the overall number of commits by a developer, and permission-based commits?

Our first goal was to calculate the ratio of a developer's permission-based commits to overall project commits. Understanding the project experience levels of who is making permissions-related project decisions is important since permissions often negatively affect a user's perception of the app [8] and carry numerous security related risks [10,22]. Using oSARA, we recorded the developer's DCR score for each added or removed permission in the app's commit history, which provided us with a ratio of a developer's permissions-related commits to overall project commits. In our analysis, we were careful to count only altered permissions in an app's commit history. If a developer did not change the app's requested permissions, this commit was not included in our permissions-related analysis since it did not include a permissions-related alteration. However, this commit would be included in our count of the overall developer commits since it did involve other alterations to the project. We recorded the average DCR mean, mode, median, and standard deviation for all analyzed apps. The results of this analysis is shown in Table 1.

Table 1: Developer DCR Statistics

Permission Action	Mean	Mode	Median	Std
Add	.52	.84	.45	.333
Remove	.524	.84	.5	.335
Total	0.521	0.84	.5	.334

Although the mean of the DCR values are centered around .52, the large standard deviation of .33 demonstrates a significant variation of the DCR levels among all of the developers. In order to better understand this large range, we next examined the DCR level of committers at a much more granular level. Our first step was to group all developers into five groups based on their percentage of commits they were responsible for in an individual project. For each of these groups, we determined the average number of permissions both added and removed for each developer in this group. The results of this analysis is shown in Figure 2.

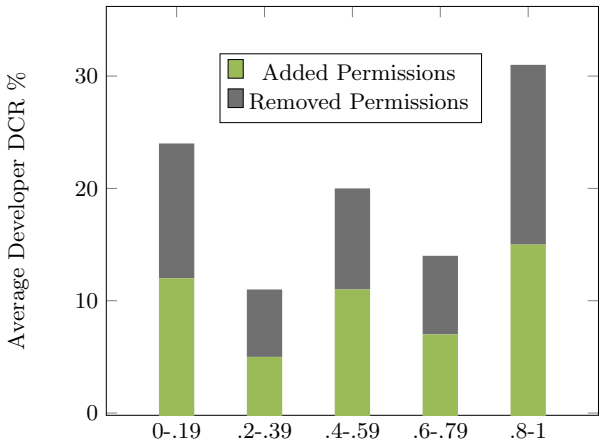

Figure 2: Developer Commit Ranges

This demonstrates that a surprisingly large number of authors with low amounts of project based commits performed a large number of permission-related commits. For example, authors who performed less than 20% of an app's total commits performed 24% of a project's permission-related commits. Permissions-related commits for authors who had significantly more project related commits was higher, but still not proportional with the number of overall project commits. We also found that developers in each group had approximately the same likelihood of adding or removing permissions.

We next used the Spearmen, Kendall and Pearson correlation metrics to determine the correlation strength between a developer's total project commits, and permission-based commits. The results of our analysis is shown in Table 2. When using these correlation metrics, the correlation coefficient will vary between -1 and +1. As the value of the coefficient approaches ±1, there is more of a monotonic, or perfect degree of association between the evaluated values. The relationship between the evaluated values becomes weaker as the correlation coefficient approaches 0. Our findings demonstrate an extremely weak association between the total number of application commits, and permissions-related commits. This further indicates that developers with a low number of project commits were making a disproportionally high number of permissions-related commits.

Table 2: Correlation of Total Commits and permission-based Commits

Spearman	Kendall	Pearson
0.044	0.025	0.067

Analysis: The implications for these results are important since the proper use of permissions is crucial for a variety of security and functional reasons [8, 10, 22]. A key question is why developers with such a low number of overall project commits are making a disproportionately high number of permissions-based decisions. There are several possible explanations for this which should be further explored. One possibility is that developers with less project commits will have less overall project knowledge, and will mistakenly believe that permissions should be added or removed from the project. A contrarian view would suggest that 'permission experts' perform a large number of permission-related changes to apps, but do not make a significant number of other alterations.

RQ2: Is there a correlation between the ratio of developer permission-based commits to overall project commits, and user ratings?

Users of an app frequently provide feedback in the form of ratings. Apps which regularly receive poor ratings will be low-rated, apps, while those who receive more favorable feedback will become high-rated apps. Potential app users will frequently use these ratings to determine if they will choose to download the app, so having high-rated apps are of immense importance for app developers [4, 12, 17].

We found that low percentage committers were making a large amount of permission changes, so our next goal was to examine any effects this may have on user ratings. From our collected data, we performed Spearman, Kendall and Pearson correlations to determine if there was a relationship between the ratio of developer permission-based commits to overall project commits, and user ratings for an app. The results of this analysis are shown in Table 3.

Table 3: DCR Score and User Rating Correlation

Spearman	Kendall	Pearson
0.086	-0.059	-0.0969

Analysis: These values indicate very weak relationships between the ratio of developer permission-based commits to overall project commits, and user ratings. Although developers with a low number of project commits are making a disproportionately high number of permissions-related commits, this does not appear to have a strong effect on user ratings (perception) of an app. There are several possible reasons to explain this weak correlation. One reason is that several previous works [11, 16] have argued that users do not pay attention to an app's permissions, so that the permissions and who added them, would not affect the ratings of an app. Contrary to these works, other research has argued that users do indeed pay a significant amount of attention to an app's permissions, and that it may affect their perception of an app [8, 20].

RQ3: At what development stages are permissions being added to Android projects?

Our final research question was to understand the rate that permissions are being added and removed from Android apps. We began by selecting apps with at least 120 commits to their AndroidManifest.xml file, which did not need to contain alterations to the requested app permissions, only changes to any part of the file. We selected 120 commits since we had at least 25 different apps with this number of AndroidManifest.xml commits. For commits 1-120 of the AndroidManifest.xml file, we determined the average number of permissions requested in each group. The results are shown in Figure 3.

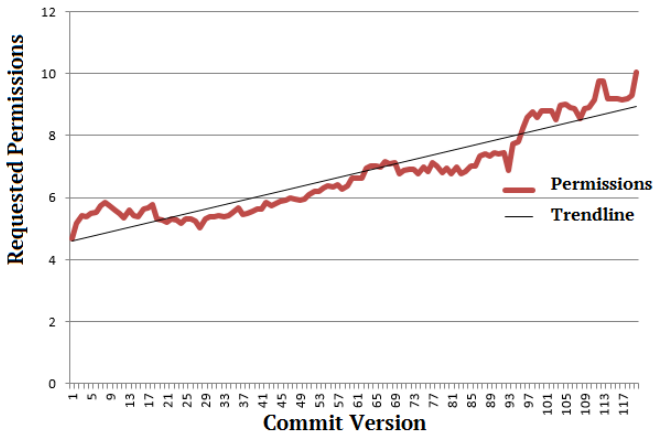

Figure 3: Rate of Added Permissions

On average, apps began with 4.68 permissions in their initial commit, and grew to requesting 10.05 permissions in commit 120. Our findings demonstrate that requested permissions are growing at a reasonably steady rate throughout an app's life cycle. Permissions change with every new Android API release and the number of available permissions has steadily grown since Android's inception. However, the rate of per-

missions being added to apps seems to far outpace the rate that new Android permissions are being created [3].

Analysis: These results indicate that permission-related decisions are made at a fairly steady rate throughout the development of the project, which demonstrates the need for developers to be constantly vigilant about properly using permissions. As with other types of functional and security testing, proper permission usage needs to be consistently checked throughout the development of an app.

6. PUBLIC DATA SET & TOOL

We have shared our primary discoveries, data, and all created software for this study on our project website at: **http://www.Hidden.com**.

6.1 Data

Our data set contains all raw and processed project data including all extracted *AndroidManifest.xml* files, version control information, permission changes and permission history. An overview of some of the publicly accessible data is shown in Table 4. All data is available for external use and is available in an SQLite database.

Table 4: Overview of Public Data Set

Value	Count
Total Projects	1,179
Android Manifest files	26,450
Permission Changes	35,769
Total project commits	435, 680

This website not only contains the raw and processed project data, but also contains a suite of web based tools that other researchers may use in their work including the ability to query the data set on the web, and several customizable and pre-built reports.

6.2 oSARA Tool

We created Open Source Android Repository Analyzer (oSARA), to assist with crucial data extraction and analysis for this project. oSARA begins by extracting commit information from the collected git repository including who the committer was, when the commit was made, and the commit message. The tool then extracts all committed AndroidManifest.xml files from the version control history and records all modified permissions in these files. Using thise collected information, oSARA then determines all altered permissions, who made the alterations, and when they were made.

This tool and relevant documentation is available in a public version control system which is available on our project website. We encourage others to use this tool to not only replicate our study, but to build upon it to conduct their own research as well.

7. LIMITATIONS & 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.

Previous work has analyzed apps through the examination of version control repositories and many studies have integrated developer interviews in their analysis. Our work should be expanded to include more developer interviews and more qualitative information. Further work could also 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 Stowaway [10], PScout [5], FindBugs [1], or TaintDroid [9]

Android apps often suffer from permissions-misuse where apps often request too few, or too many permissions [10,13]. Future work may be done to analyze if the DCR score of an permission has a higher rate of being misused. This analysis may be conducted using a permissions analysis tool such as PScout [5]. Future work could be conducted to 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.

We looked for a correlation between who made permissions based app alterations and the app's user rating. This may be an imprecise correlation since evaluating apps through user ratings is a difficult task as there are numerous security and quality metrics which may affect a user's perception of the app. Some studies have found that permissions significantly affect a user's perception of an app [20], while other studies have demonstrated that permissions play a much more insignificant role in a user's perception of an app and that there are numerous other factors which play a much more significant role [11, 16].

8. CONCLUSION

Summary: We examined 1,179 open source Android apps from the F-Droid repository to better understand when, why and how permissions are added to open source Android apps. We also checked for a correlation between the number of permissions and total app commits by a developer, and user ratings.

Findings: We found that: I) Developers with a low number of project based commits make a disproportionately high number of permissions-related commits. II) The user rating of an app is not affected by the number of permissions added by low or high-committers. III) Permissions are added at a reasonably consistent rate to an app during its development.

9. REFERENCES

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

[2] Number of apps available in leading app stores as of november 2015. http://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/.

[3] System permissions. http://developer.android.com/guide/topics/security/permissions.html.

[4] A. Al-Subaihin, A. Finkelstein, M. Harman, Y. Jia, W. Martin, F. Sarro, and Y. Zhang. App store mining and analysis. In *Proceedings of the 3rd International Workshop on Software Development Lifecycle for Mobile*, pages 1–2. ACM, 2015.

[5] 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.

[6] 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.

[7] J. Edwards. iphone lost market share to android in every major market except one. http://www.businessinsider.com/apple-ios-v-android-market-share-2016-1?r=UK&IR=T, January 2016.

[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] 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.

[11] A. P. Felt, E. Ha, S. Egelman, A. Haney, E. Chin, and D. Wagner. Android permissions: User attention, comprehension, and behavior. In *Proceedings of the Eighth Symposium on Usable Privacy and Security*, SOUPS '12, pages 3:1–3:14, New York, NY, USA, 2012. ACM.

[12] B. Fu, J. Lin, L. Li, C. Faloutsos, J. Hong, and N. Sadeh. Why people hate your app: Making sense of user feedback in a mobile app store. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1276–1284. ACM, 2013.

[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] J. Gui, S. Mcilroy, M. Nagappan, and W. G. J. Halfond. Truth in advertising: The hidden cost of mobile ads for software developers. In *Proceedings of the 37th International Conference on Software Engineering - Volume 1*, ICSE '15, pages 100–110, Piscataway, NJ, USA, 2015. IEEE Press.

[15] M. Harman, Y. Jia, and Y. Zhang. App store mining and analysis: Msr for app stores. In *Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on*, pages 108–111, June 2012.

[16] P. G. Kelley, S. Consolvo, L. F. Cranor, J. Jung, N. Sadeh, and D. Wetherall. A conundrum of permissions: Installing applications on an android smartphone. In *Proceedings of the 16th International Conference on Financial Cryptography and Data Security*, FC'12, pages 68–79, Berlin, Heidelberg, 2012. Springer-Verlag.

[17] H. Khalid, M. Nagappan, and A. E. Hassan. Examining the relationship between findbugs warnings and end user ratings: A case study on 10,000 android apps. In *IEEE Software Journal*, 2014.

[18] H. Khalid, M. Nagappan, E. Shihab, and A. E. Hassan. Prioritizing the devices to test your app on: A case study of android game apps. In *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2014, pages 610–620, New York, NY, USA, 2014. ACM.

[19] 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.

[20] J. Lin, S. Amini, J. I. Hong, N. Sadeh, J. Lindqvist, and J. Zhang. Expectation and purpose: Understanding users' mental models of mobile app privacy through crowdsourcing. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, pages 501–510, New York, NY, USA, 2012. ACM.

[21] M. Linares-Vásquez, G. Bavota, C. Bernal-Cárdenas, M. Di Penta, R. Oliveto, and D. Poshyvanyk. Api change and fault proneness: A threat to the success of android apps. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2013, pages 477–487, New York, NY, USA, 2013. ACM.

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

[23] 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.