Examining the Relationship between Security Metrics and User Ratings of Mobile Apps: A Case Study

Daniel E. Krutz, Nuthan Munaiah, Casey Klimkowsky, Shannon Trudeau, Adam Blaine, Andrew Meneely, and Sam Malachowsky
Rochester Institute of Technology, Rochester, NY, USA
{dxkvse, nm6061, cek3403, smt9020, amb8805, axmvse, samvse}@rit.edu

ABSTRACT

The success or failure of a mobile application ('app') is largely determined by user ratings. Users frequently make their app choices based on the ratings of apps in comparison with similar apps. Users also expect apps to continually provide new features while maintaining quality, or the ratings drop. At the same time apps must also be secure, but is there a historical trade-off between security and ratings? Or are app store ratings a more all-encompassing measure of product maturity? We collected and compared several security related metrics from 38,466 Android apps in the Google Play store. Using several security-based static analysis tools, we compared an app's rate of permissions misuse, number of requested permissions, and Androrisk score against its user rating. Based on our evaluations, we found that there was no significant correlation between an app's user ratings, and its security risks.

[Dan says: what more can we add here?] **[make sure to update this based on our results]**

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection;

Keywords

Android, User Ratings, Security

1. INTRODUCTION

Android is the world's most popular mobile OS [**?**] with over 1.9 million apps available from Google Play alone [**?**]. The success of an Android app is largely dependent on user ratings presented in this digital storefront; users expect apps to continuously provide new features, threatening poor app store reviews and low-ratings if this expectation is not met [**?**]. In keeping up with the rapid progress of mobile technology, developers also frequently find themselves needing to update their app's dependencies [**?**]. Most importantly, apps are a crucial entry point into our digital lives, and therefore must be secure.

At a glance, one may assume that the challenge of security and customer satisfaction are trade-offs, since if developers focus on new features to keep the ratings up, security testing on an ever-increasing codebase may slip. New security-inspired features may also be perceived by users as cumbersome or unnecessary, leading to lower ratings. Even a vulnerability in a dependency can be detrimental to users, yet many developers may not have the resources or commitment needed to thoroughly inspect a third-party framework for security concerns. Experts have even warned that security trade-offs with other properties such as usability and performance are considered universal [**?**].

But is this trade-off historically true in the case of mobile apps? Empirically, do mobile apps with higher ratings have more potential security risks? Or, do app store ratings represent a more all-encompassing measure of customer experience which indicates maturity in all of the properties of an app, with security being just one aspect? These questions motivated us to empirically examine the relationship between user ratings and security. To measure potential security risks, we use automated static analysis tools specifically tailored to the Android platform. While far from a comprehensive security audit, the static analysis tools provide a broad and consistent measure of basic security flaws that might plague Android apps. To measure user rating, we extracted the user ratings of 38,466 Android apps from the Google Play app store.

The objective of this study is to investigate the relationship between potential security risks and customer satisfaction by empirically evaluating Android apps with static analysis tools. Specifically, our research question is: *Are user ratings correlated with low potential security risks and security permissions, or do apps with higher ratings have more security risks?*

We found that there was no meaningful correlation between the user rating of an app, and several collected security metrics. This indicates that the app's security metrics have no direct impact on an app's ratings either because the user has no ability to comprehend the app's actual security levels, or because they do not see....**[make sure to update this based on our results]**

The rest of the paper is organized as follows: In Section **??** we present the design of our case study, where we explain what tools we use, what data we collect and how we collect it. In Section **??**, we present the results of our case study, and in Section **??**, we discuss related work. In Section **??**, we present the threats to validity for our study, and Section **??** concludes the paper based on our findings. **[reorder this]**

2. RELATED WORK

[Dan says: Proof entire section, especially the bottom since I rewrote much of it]

There has been a substantial amount of previous research analyzing the effects of permissions on the user's perception of the app. Lin et al. [**?**] 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 gen-

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

erally felt uncomfortable and may even delete applications when they did not understand why it requested a permission they deemed unnecessary. Egelman et al. [**?**] 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 [**?**]. Kelley et al. [**?**] 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.

Stevens *et al.* [**?**] 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.

App ratings have demonstrated their importance in other areas of research as well. Harman et al. [**?**] found a strong correlation between the rating and the number of app downloads. Linares-Vasquez et al. [**?**] found that change and fault-proneness of the APIs used by the apps negatively impacts their user ratings. Khalid et al. [**?**] 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.

There has also been a substantial amount of work regarding the risks of over-permissions in Android apps. Felt et al. [**?**] discussed the dangers of app over-permissions including unnecessary permissions warnings and exposure to various bugs and vulnerabilities. The study also examined 940 Android apps and found that about 33% of them contained over-permissions.

Grace et al. [**?**] conducted work on permissions probing, which is when a 3rd party app attempts to use a permission in the hope that the attached app has requested them from the user. This is often done to collect and transmit potentially sensitive information which should not be normally available to the 3rd party app. They found that more than half of all ad libraries try to probe for open permissions. This could potentially 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.

Peng et al. [**?**] introduced the idea of risk scoring and risk ranking in Android apps in order to effectively conduct risk communication in Android apps. This work proposed the use of probabilistic generative models for risk scoring with the goal of accurately and effectively detecting malware. Gorla et al. [**?**] conducted research to examine if an app's description matched its functionality in an attempt to discover malware. Their system was able to correctly flag 56% of malware, without the use of any known malware patterns. Tian et al. [**?**] distinguished high and low-rated apps using various metrics such as code complexity, API quality, and API dependencies. In their case study which used 1,492 low-rated and high-rated apps, they found that an app's most influential factors included the size of the app, number of promotional images, target SDK version and the number of promotional images on the app store page are the most significant factors in determining an app's rating.

3. STUDY DESIGN

We first collected a variety of apps from Google Play using a modified collection tool and then analyzed them using several well-known Android static analysis tools. An overview of our collection and analysis process is shown in Figure **??**. We will next describe our data collection, selection and analysis process.

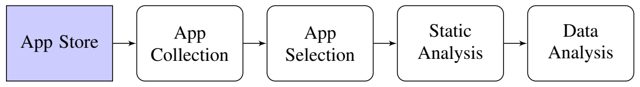

Figure 1: App Collection and Analysis Process

3.1 Data Collection & Selection

We collected over 70,000 apps from the Google Play store with a custom-built collector, which uses *Scrapy*[1] as a foundation. In order to gather a diverse set of apps, all apps were randomly pulled from the Google Play store. We chose to pull from Google Play since it is the most popular source of Android apps [**?**] and was able to provide other app related information such as the genre, user rating, and number of downloads, which we stored in a SQLite database.

In order to include only reasonably popular apps in our study, we excluded all apps with less than 10,000 downloads from our analysis, which left us with 38,466 apps. The minimum rating of our collected apps is 1.4, and the maximum is 5 stars. The average rating of these apps is 3.99 and the median is 4.1. Our collection includes apps from 41 different app categories with 'Tools' apps accounting for the highest number and Music apps accounting for the least.

In order to create a group of low-rated, and high-rated apps for comparison, we next sorted apps based on their ratings, using a process similar to that Tian et al [**?**]. To create a set of low-rated apps, we selected the bottom 10% of rated apps in our study, while our high-rated apps were comprised of the top 10% rated apps in our study. This created a set of 3,847 apps in both our low-rated and high-rated data sets. The median rating of the low-rated apps is 3.2, while it is 4.6 for high-rated apps. The median number of downloads for low-rated apps is 10,000 and is 100,000 for high-rated apps.

[Dan says: Nuthan: Can you create a boxplot of average user ratings like in Mei's [?]]

3.2 Static Analysis Tools

The next phase was to analyze the apps for potential security risks and permissions issues. In addition to using APKParser[2] to collect an app's requested permissions, we used two open-source static analysis tools in our study: Stowaway [**?**] and Androrisk[3]. Stowaway evaluates the app for permission misuse, and Androrisk determines the risk vulnerability level.

We selected Stowaway for determining permission misuse since it is able to state the specific permissions that are causing permissions gaps, while using a static analysis-based approach that did not require it to be run on an Android device or through an emulator. Stowaway has also demonstrated its effectiveness in existing research [**?**, **?**, **?**]. Stowaway extracted the number of under and over-permissions that are present in each app. This tool is comprised of two parts: API calls made by the app are determined using a static

[1]http://scrapy.org
[2]https://github.com/joakime/android-apk-parser
[3]https://code.google.com/p/androguard/

analysis tool, and the permissions needed for each API are determined using a permissions map. Similar to previous work [**?**], we made slight modifications to Stowaway to accommodate our process and stay current with updated Android permissions.

Androrisk determines the security risk level of an application by examining several criteria. The first set is the presence of permissions which are deemed to be more dangerous, such as the ability to access the internet, manipulate SMS messages, or to make a payment. The second is the presence of more dangerous sets of functionality in the app including utilizing a shared library, use of cryptographic functions, and the presence of the reflection API.

We chose Androrisk for several reasons. The first is that Androrisk is part of the Androguard library, which has already been used in a variety of existing research [**?**, **?**, **?**]. Since it is a static analysis-based vulnerability detection tool, Androrisk was quickly able to effectively determine the risk level of a large number of downloaded apps.

APKParser was used to collect a variety of information about the app, including its requested permissions. The primary difference between requested permissions and over-permissions is that requested permissions are merely those that the app asks to use, not taking into consideration whether the app actually needs them or not.

4. RESULTS

[update all of this]

In this section, we discuss the motivation, approach, and findings for our research question. We then provide a more detailed discussion of our results.

[add in scatter plot] *[Dan says: Nuthan: Can you add the scatter plot, along with an analysis of the scatter plot?]*

RQ: Do apps with low-rated have more security risks?

[check for repeated section in motivation] **Motivation:** Android developers operate under a permission-based system where apps must be granted access to various functionality in order to be used. When an Android app is created, developers must explicitly declare which permissions the application will require [**?**], such as the ability to write to the calendar, send SMS messages, or access the GPS. If an app attempts to perform an operation for which it does not have permission, a *SecurityException* will be thrown. For Android versions 1-5, the user is asked to accept or reject requested permissions when installing the app. Once installed, the developer cannot remotely modify the permissions without releasing a new version of the app for installation [**?**]. The user will then be prompted to accept whatever new permissions have been requested in the update. Beginning with Android 6.0, developers may ask the user to accept permissions at runtime, instead of only during the installation or upgrade process. **[check on what is being repeated]**

A basic principle of software security is the *principle of least privilege*. In the context of mobile apps it translates to granting the minimum number of permissions that an app needs to properly function [**?**]. Granting more permissions than the app needs creates unnecessary security vulnerabilities since vulnerabilities in the app (or malware) could use these extra permissions for malicious reasons. Additionally, eliminating unnecessary permissions limits potential issues due to non-malicious developer errors and reduces the app's attack surface [**?**].

Unfortunately, developers often request more permissions than they actually need, as there is no built in verification system to ensure that they are only requesting the permissions their app actually uses [**?**]. Developers misuse permissions for a variety of reasons including lack of understanding about the permissions and inadequate community support [**?**]. 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 a setting for which the app could fail because it was not given the proper permissions. Over-permissions are considered security risks and under-permissions are considered quality risks. Although an app may require permissions for numerous legitimate purposes, more permissions increases an app's attack surface, making it vulnerable to outside sources [**?**, **?**]. As an example, permissions may be unknowingly misused in a variety of ways by 3rd party libraries or even associated ad networks, potentially collecting and transmitting potentially sensitive user data [**?**, **?**].

Approach: In order to answer our research question, we used Stowaway, Androrisk, and APKParser to check if the values of permission and risk-based metrics are different in low and high-rated apps. Our null hypothesis is that there is no difference in the distribution of the security metrics between the low and high-rated apps. Our alternate hypothesis is that low and high-rated apps have different distributions for each of the security related metrics. We use the one tailed Mann Whitney U (MWU) test for the hypothesis testing, since it is non-parametric and we can find out if the low-rated apps indeed have higher or lower values for each of the security metrics. In our analysis, we used an α-value of .05 to determine if the null hypothesis can be rejected or not. We represent $p > 0.05$ with a '-' in our results. We next measured the strength of association between our collected security metrics using the Spearman rho correlation metric. To create a fair comparison, we normalized all security metrics by LOC before using them in the MWU tests and correlation analyses.

Findings: Table **??** indicates that low-rated apps typically have more under and over-permissions, a higher Androrisk score, and a larger number of requested permissions. We next sought to determine if these same results held true for apps with similar functionality, so we next separated the results by app category. We found three app categories where we had at least 100 results for both low and high-rated apps. Our findings show that low-rated apps are not using many of the permissions they ask for, which is quite dangerous as this leaves the software much more prone to vulnerabilities. The results also indicate that low-rated apps contain more under-permissions, which implies that there may be components trying to conduct activities for which they do not have appropriate permission. Such a problem not only indicates quality issues, but could also indicate that these apps may contain functionality which is probing for open or available permissions**[info to back this up]** [**?**, **?**].

We used the Spearman rho metric to determine the strength of correlation between our observed security metrics. Overall, we found a very weak correlation between user ratings and our observed security metrics, with the one exception being the number of requested permissions, which was still a fairly weak correlation.

To answer our primary research question, our data suggests that:

Low-rated apps have higher security risk metrics (more risk) than high-rated apps, but with very weak correlations. ***[update]***

4.1 Discussion

[update all of this]

In this section, we examine and provide some possible explanations for our findings.

What are some possible explanations for our findings? There are several possible explanations as to why low-rated apps suffer from more security vulnerabilities. The presence of vulnerability defects may indicate the existence of other, user observable issues

Table 1: MWU & Spearman Analysis Results[update the data]

		MWU		
		Greater In		
Category	**Analysis**	**Low**	**High**	**p-value**
All	Permissions			X
	Over-Permissions	X ✓X		X
	Under-Permissions			X
	Androrisk Score			X
Tools	Permissions			X
	Over-Permissions	X ✓XX		X
	Under-Permissions			X
	Androrisk Score			X
Entertainment	Permissions			X
	Over-Permissions	X ✓XX		X
	Under-Permissions			X
	Androrisk Score			X
Education	Permissions			X
	Over-Permissions	X ✓XX		X
	Under-Permissions			X
	Androrisk Score			X
Personalization	Permissions			X
	Over-Permissions	X ✓XX		X
	Under-Permissions			X
	Androrisk Score			X
Puzzle	Permissions			X
	Over-Permissions	X ✓XX		X
	Under-Permissions			X
	Androrisk Score			X

which may impact the rating of the app. Users may also notice the larger number of permission requests in low-rated apps, which may lead to privacy concerns, and therefore lower reviews [?]. Under-permissions may cause an app to crash when it attempts to utilize the permission it has not been granted[cite]. While the user will be very unlikely to know that the crash was due to an under-permission, the encountered defect could still impact their review of the app.

The only reasonably strong correlation we found was with the number of permissions that were requested by low-rated apps. While users are unlikely to perceive an app's permission gap, or Androrisk score, they do typically comprehend the number of permissions requested by the app since they are required to approve all 'dangerous' permissions requested by the app.

Recent studies have examined the effects of permission requests on users. While some findings have indicated negative effects of more permissions requests on users [?, ?], other research has shown that users typically ignore permissions, and what an app requests has no affect user's perception of an app [?, ?]. Since previous research is largely contradictory, it comes to no large surprise that we found only a weak correlation between the number of requested permissions and user ratings.

Does it mean that low-rated apps are less secure? Possibly. A primary way that over-permissions make an application less secure is that they increase the attack surface [?, ?], thus making it more susceptible to malware and other vulnerabilities. However, identifying actual vulnerabilities using any static analysis tool is a difficult task, as these tools only look for potential vulnerabilities [?]. Our findings may only conclude that low-rated apps have a higher rate of *potential* vulnerabilities, and request more permissions in comparison to high-rated apps. Additionally, we found only a weak correlation between the rating of an app and potential vulnerabilities.

5. LIMITATIONS & FUTURE WORK

While we feel that our findings are profound, they are not without their limitations. Although Google Play is the largest Android source, it is not the only location for attaining Android apps. Alternatives include the Amazon app store[4], F-Droid[5], and many other sources; other studies may choose to include apps from these locations. Additionally, we only examined a total of 38,466 apps, which is a small minority of the over 1.9 million available Android apps [?]. However, given that this is a random sample we believe that it is representative of the Android application population. Future work could be done to include paid apps in a similar analysis since we only examined free apps.

While static analysis tools have demonstrated their value in numerous previous works [?, ?], no static analysis tool is perfect and often inherently contains limitations [?]. Although Stowaway is a powerful static analysis tool which has been used in previous research [?, ?, ?], it does suffer from drawbacks; Stowaway's own authors state that the tool only achieves 85% code coverage [?], so the misused permissions reported by this tool are imperfect. Additionally, any reported vulnerabilities by a static analysis tool should be deemed as *possible* vulnerabilities, not actual vulnerabilities, since the only way of identifying an actual vulnerability is through manual analysis and verification [?].

Identifying possible vulnerabilities or security risks is extremely difficult, and like any static analysis tool, Androrisk is only capable of making educated observations about the risk level of an app. More substantial risk assessments require a far more analysis,

[4]http://www.amazon.com/mobile-apps/b?node=2350149011
[5]https://f-droid.org/

which would likely include a manual investigation of the app. Due to the large number of examined apps in our study, this thorough level of analysis was not practical. Even with almost certain imperfections, we believe that Androrisk was a good choice due to its ability to quickly analyze apps and its use in existing research [?].

In our evaluation, we only measured the quantitative user ratings of apps. Future work could examine the text, looking for security or permissions complaints in apps which have more possible vulnerabilities, more permissions, or more over-permissions. A similar analysis to previous works [?, ?] may be conducted which could include a keyword frequency analysis or other techniques for user review.

We only analyzed pre-Android 6.0 apps since relatively few Android 6.0 apps were available for analysis. Android 6.0 received a massive permissions overhaul and future work could examine how this new release affects developers use of permissions and how customers perceive these apps.

Only measuring one quality aspect of the app. other factors could be include such as defects, and code quality

[Dan says: many users will never actually see vulnerabilities, so how could it affect ratings?]

[overprivs are not always bad]

6. CONCLUSION

Summary: The goal of our research was to determine if low-rated apps suffered from more possible security vulnerabilities than high-rated apps. We statically analyzed 38,466 Android apps for security threats primarily using several static analysis tools. We collected requested permissions using APKParser, evaluated apps for under & over-permissions using Stowaway, and implemented Androrisk for a more general security risk assessment.**[add a bit more]**

Findings: According to several static analysis tools, we found that low-rated apps had a greater rate of having under and over-permissions, and had a higher Androrisk score. While statistically significant, the correlations are very weak except in the case of the number of an app's requested permissions. Our results indicate that apps with more security risks may suffer in other dimensions as well. **[make sure to update this based on our results]**

7. REFERENCES

[1] Appbrain stats. http://www.appbrain.com/stats/number-of-android-apps.

[2] *Building Secure Software: How to Avoid Security Problems the Right Way*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[3] A. Atzeni, T. Su, M. Baltatu, R. D'Alessandro, and G. Pessiva. How dangerous is your android app?: An evaluation methodology. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, MOBIQUITOUS '14, pages 130–139, ICST, Brussels, Belgium, Belgium, 2014. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[4] B. Chess and G. McGraw. Static analysis for security. *IEEE Security & Privacy*, (6):76–79, 2004.

[5] 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.

[6] M. Egele, D. Brumley, Y. Fratantonio, and C. Kruegel. An empirical study of cryptographic misuse in android applications. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer Communications Security*, CCS '13, pages 73–84, New York, NY, USA, 2013. ACM.

[7] 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.

[8] 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.

[9] 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.

[10] 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*, KDD '13, pages 1276–1284, New York, NY, USA, 2013. ACM.

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

[12] J. Giggs. The ultimate app store list. http://www.businessofapps.com/the-ultimate-app-store-list/, Feb. 2015.

[13] A. Gorla, I. Tavecchia, F. Gross, and A. Zeller. Checking app behavior against app descriptions. In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 1025–1035, New York, NY, USA, 2014. ACM.

[14] 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.

[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] J. Jeon, K. K. Micinski, J. A. Vaughan, N. Reddy, Y. Zhu, J. S. Foster, and T. Millstein. Dr. android and mr. hide: Fine-grained security policies on unmodified android. 2011.

[17] 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.

[18] 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.

[19] H. Khalid, E. Shihab, M. Nagappan, and A. Hassan. What do mobile app users complain about? a study on free ios apps. *IEEE Software*, 99(PrePrints):1, 2014.

[20] D. E. Krutz, M. Mirakhorli, S. A. Malachowsky, A. Ruiz, J. Peterson, A. Filipski, and J. Smith. A dataset of open-source android applications. In *Mining Software Repositories (MSR), 2015 IEEE/ACM 12th Working Conference on*, pages 522–525. IEEE, 2015.

[21] 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.

[22] 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.

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

[24] P. Pearce, A. P. Felt, G. Nunez, and D. Wagner. Addroid: Privilege separation for applications and advertisers in android. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '12, pages 71–72, New York, NY, USA, 2012.

[25] H. Peng, C. Gates, B. Sarma, N. Li, Y. Qi, R. Potharaju, C. Nita-Rotaru, and I. Molloy. Using probabilistic generative models for ranking risks of android apps. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 241–252, New York, NY, USA, 2012. ACM.

[26] J. H. Saltzer and M. D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

[27] K. Shaerpour, A. Dehghantanha, and R. Mahmod. Trends in android malware detection. *Journal of Digital Forensics, Security & Law*, 8(3), 2013.

[28] R. Stevens, J. Ganz, V. Filkov, P. Devanbu, and H. Chen. Asking for (and about) permissions used by android apps. In *Proceedings of the 10th Working Conference on Mining Software Repositories*, MSR '13, pages 31–40, Piscataway, NJ, USA, 2013. IEEE Press.

[29] R. Stevens, C. Gibler, J. Crussell, J. Erickson, and H. Chen. Investigating user privacy in android ad libraries.

[30] M. D. Syer, M. Nagappan, A. E. Hassan, and B. Adams. Revisiting prior empirical findings for mobile apps: An empirical case study on the 15 most popular open-source android apps. In *Proceedings of the 2013 Conference of the Center for Advanced Studies on Collaborative Research*, CASCON '13, pages 283–297, Riverton, NJ, USA, 2013.

[31] Y. Tian, M. Nagappan, D. Lo, and A. E. Hassan. What are the characteristics of high-rated apps? a case study on free android applications. In *Software Maintenance and Evolution (ICSME), 2015 IEEE International Conference on*, pages 301–310. IEEE, 2015.

[32] T. Vidas, J. Tan, J. Nahata, C. L. Tan, N. Christin, and P. Tague. A5: Automated analysis of adversarial android applications. In *Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones &#38; Mobile Devices*, SPSM '14, pages 39–50, New York, NY, USA, 2014. ACM.