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

Daniel E. Krutz, Nuthan Munaiah, Andrew Meneely, and Samuel A. Malachowsky
{dxkvse,nm6061,axmvse,samvse}@rit.edu
Department of Software Engineering
Rochester Institute of Technology, Rochester, NY, USA

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, often competing 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 used static analysis tools to collect security-related metrics in 38,466 Android apps from the Google Play store. We compared the rate of an app's permission misuse, number of requested permissions, and Androrisk score, against its user rating.

We found that high-rated apps have statistically significantly higher security risk metrics than low-rated apps. However, the correlations are weak. This result supports the conventional wisdom that users are not factoring security risks into their ratings in a meaningful way. This could be due to several reasons including users not placing much emphasis on security, or that the typical user is unable to gauge the security risk level of the apps they use everyday.

CCS Concepts

•**Software and its engineering** → *Software design trade-offs;*

Keywords

Android, User Ratings, Security

1. INTRODUCTION

Android is the world's most popular mobile OS [6] with over 1.9 million apps available from Google Play alone [1]. The success of an Android app is largely dependent on the user ratings available on the digital storefront. Users expect apps to continuously provide new features, threatening poor app store reviews and low ratings if their expectations are not met [18]. 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 [2]. 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. We extracted the user ratings of 38,466 Android apps randomly collected 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, we address the following research question:

RQ Correlation Does average user rating of mobile apps correlate with security risk?

We found a weak correlation between the security-related risk metrics collected from an app and its user rating. However, the separation of the apps into two populations—low-rated and high-rated—revealed a statistically significant association between three of the four security-related risk metrics and user rating with high-rated apps being more likely to have higher values for the security-related risk metrics.

The remainder of this paper is organized as follows: In Section 2, we discuss related work and in Section 3 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 4, we present the results of our case study and in Section 5, we

WAMA'16, November 14, 2016, Seattle, WA, USA
© 2016 ACM. 978-1-4503-4398-5/16/11...$15.00
http://dx.doi.org/10.1145/2993259.2993260

discuss the threats to validity for our study. Section 6 concludes the paper based on our findings.

2. RELATED WORK

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. [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 apps 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 app, 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 [10]. 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.

Stevens et al. [26] 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. [14] found a strong correlation between the rating and the number of app downloads. Linares-Vasquez et al. [21] found that change and 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', 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.

There has also been a substantial amount of work regarding the risks of over-permissions in Android apps. Felt et al. [9] 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. [13] 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 advertisement (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.

Tian et al. [27] distinguished high and low-rated apps using various metrics such as code complexity, API quality, and API dependencies. In their case study of 1,492 low-rated and high-rated apps, they found the size of the app, number of promotional images on the app store page, and the target SDK version to be the most influential factors affecting the user rating of an app.

3. STUDY DESIGN

We first collected a variety of apps from Google Play using a custom-built 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 1. We will next describe our data collection, selection and analysis process.

Figure 1: App Collection and Analysis Process

3.1 App 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 [12] and was able to provide other app related information such as the app category, 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, leaving us with 38,466 apps. The minimum and maximum rating of our collected apps is 1.4 stars and 5 stars. The average rating of these apps is 3.99 stars and the median is 4.1 stars. Our collection includes apps from 41 different app categories with 'Tools' and 'Music' categories accounting for the highest and lowest number of apps.

To create a group of low-rated, and high-rated apps for comparison, we next sorted the apps in descending order based on their average user rating. Using an approach similar to that used by Tian et al. [27], we separated the sorted list of apps into two groups :1. *high-rated apps*, which comprised of apps from the top 10% of the sorted list and 2. *low-rated apps*, which comprised of apps from the bottom 10% of the sorted list. Since our data set contained 38,466 apps, the low and high-rated app groups contained 3,846 apps each. Figure 2 shows the distribution of number of downloads and average user rating of apps in these two groups. 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. The number of downloads metric is the minimum number from the download range reported on the Google Play store.

3.2 Analysis

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

[1] `http://scrapy.org`

[2] `https://github.com/joakime/android-apk-parser`

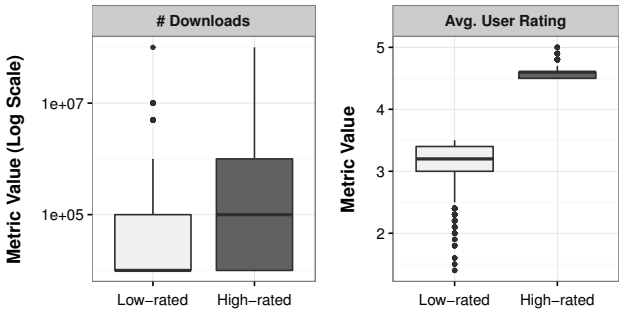

Figure 2: Distribution of Number of Downloads and Avg. User Ratings of the Low-rated and High-rated Apps

two open-source static analysis tools in our study: Stowaway [9] and Androrisk[3]. Stowaway evaluates the app for permission misuse, and Androrisk determines the app's vulnerability risk 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 [9,23,26]. 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 analysis tool, and the permissions needed for each API are determined using a permissions map. Similar to previous work [26], 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 text 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 [3, 7, 28]. 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

In this section, we discuss the motivation, approach, and findings for our research question. A more detailed discussion of our results is presented in the Discussion Section §4.2.

[3]`https://code.google.com/p/androguard/`

4.1 RQ Correlation

Does average user rating of mobile apps correlate with security risk

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 [9], such as the ability to write to the calendar, send text messages, or access the location services. If an app attempts to perform an operation for which it does not have permission, a *SecurityException* will be thrown. For Android versions 1.0 through 5.0, the user is asked to accept or reject requested permissions when installing the app. Beginning with Android 6.0, developers may ask the user to accept permissions at runtime, instead of only during the installation or upgrade process.

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 [25]. 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 [22].

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 [9]. Developers misuse permissions for a variety of reasons including lack of understanding about the permissions and inadequate community support [26]. 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 often considered quality risks [4]. Although an app may require permissions for numerous legitimate purposes, more permissions increases an app's attack surface, making it vulnerable to outside sources [9, 22]. 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 sensitive user data [11, 13].

Approach: The approach that we used in the analysis of the data set in the context of our research question was conducted in two phases: x 1. an exploratory correlation analysis and 2. a detailed association analysis.

The preliminary exploratory analysis of the relationship between the average user rating of an app and the security-related metrics collected from it involved a qualitative and quantitative correlation test. We used hexagon scatter plots to qualitatively assess if there is a decipherable pattern in the relationship between average user rating and one of the four security-related risk metrics. The qualitative analysis was then supported with a quantitative analysis using the Spearman Rank Correlation Coefficient test.

Figure 3 shows the scatter plots of the average user rating of an app versus each of the four security-related risk metric considered in our study. The scatter plots, particularly the trend line shown in blue, reveal no decipherable *linear* relationship between the user rating and the security-

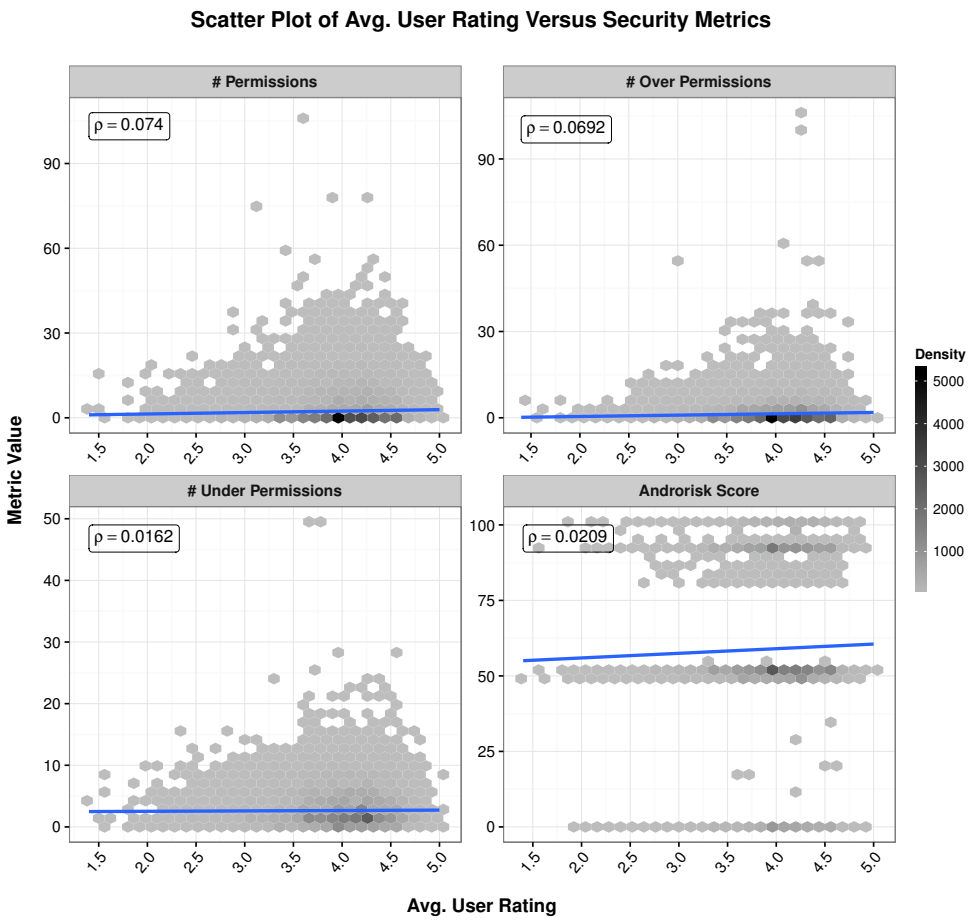

Figure 3: Scatter Plot of Avg. User Rating versus The Security-related Risk Metrics

related risk metrics. Spearman's ρ, also shown in Figure 3, strengthens the qualitative assessment, revealing a negligible correlation between the average user rating of an app and the security-related risk metrics.

Despite the negligible correlation, we attempted to analyze the correlation from an alternative perspective. In this phase of the correlation analysis, we used the two-sided Mann-Whitney U (MWU) test to assess the association between average user rating and the security-related risk metrics. In the context of the test, the null hypothesis was that low and high-rated apps have same distributions for each of the security-related metrics and the alternate hypothesis is that low- and high-rated apps have different distributions. In our analysis, we used an α-value of 0.05 to determine if the null hypothesis can be rejected or not. If the MWU test revealed the existence of a statistically significant association, we use the mean value of the metric to assess if the metric is higher or lower for low-rated apps as compared with that of the high-rated apps.

The results from the MWU test are presented in Table 1 with the statistically significant results having their p-value shown in boldface. In cases where a result was *not* statistically significant (i.e. p-value > 0.05), the comparison of the mean is shown with a - (dash) in the Table.

Findings: Table 1 indicates that, on average, high-rated apps typically have more under and over-permissions along with a higher Androrisk score. We also found that high-rated apps request more permissions but the result is not statistically significant.

We next sought to determine if these same results held true for apps with similar functionality, so we separated the results by app category choosing the five most commonly occurring groups from our collected apps. Based on this grouping, we found that apps in these categories generally conform to our overall results of high-rated apps containing more possible security vulnerabilities. Although not statistically significant in all groups, our findings show that high-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 [22, 25]. The results also indicate that high-rated apps contain more under-permissions. This problem not only indicates quality issues, but could also indicate that these apps may contain functionality which is probing for open or available permissions [11, 13].

An interesting observation was that low-rated apps in the Tools category were likely to have more permission requests that their high-rated counterparts. The permission gap, on the other hand, remained consistent with our overall observation. With the exception of the number of permissions, the results for apps in the Entertainment, Education, and Personalization categories were consistent with the overall results but were statistically insignificant.

Table 1: Analysis Results from the Mann-Whitney U Test between a Population of Low-rated and High-rated Apps

Category (# Apps)	Analysis	Greater In		p-value
		Low-rated	High-rated	
All (38,466)	Permissions	-	-	1.8535e-01
	Over-Permissions		✓	**4.0228e-19**
	Under-Permissions		✓	**8.5155e-14**
	Androrisk Score		✓	**5.1075e-07**
Tools (4,282)	Permissions	✓		**7.8488e-06**
	Over-Permissions		✓	**5.4668e-03**
	Under-Permissions		✓	**6.8640e-04**
	Androrisk Score	-	-	6.1945e-01
Entertainment (2,711)	Permissions	-	-	1.3624e-01
	Over-Permissions	-	-	1.8075e-01
	Under-Permissions		✓	**5.1918e-08**
	Androrisk Score	-	-	7.5794e-02
Education (2,381)	Permissions	-	-	3.0089e-01
	Over-Permissions	-	-	6.5054e-02
	Under-Permissions		✓	**1.0815e-04**
	Androrisk Score		✓	**1.5559e-03**
Personalization (2,049)	Permissions	-	-	3.2993e-01
	Over-Permissions	-	-	2.8604e-01
	Under-Permissions	-	-	7.5454e-01
	Androrisk Score	-	-	9.2943e-01
Puzzle (1,908)	Permissions	-	-	2.1130e-01
	Over-Permissions	-	-	6.4384e-02
	Under-Permissions		✓	**1.2566e-04**
	Androrisk Score		✓	**4.4567e-06**

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

High-rated apps have higher security risk metrics (more risk) than low-rated apps. However, there is no correlation between the rating of Android apps and the security risk metrics.

4.2 Discussion

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

What are some possible explanations for our findings?

There are several plausible explanations for high-rated apps to have a higher security risk. The first, as was evidenced in the association analysis, is that high-rated apps, on average, request a larger number of permissions, and therefore have a high chance of containing under and over privileges. However, several of the top 5 collected categories actually requested fewer permissions in high-rated apps compared to low-rated apps, while still suffering from a higher rate of under and over-privileges, along with having larger Androrisk scores. This contradicts the argument that high-rated apps suffer from more vulnerabilities merely because they request more permissions.

Our low-rated apps had a median number of 10,000 downloads, while high-rated apps in our collection had a median number of 100,000 downloads. Unfortunately, we did not examine the life cycle of apps, but one possibility is that the apps with more downloads could be older apps, and may have more downloads simply because they have been around longer. Previous research has found that apps are much more likely to add permissions, than remove them with each version update [29].

Our static analysis tools examined the entire app, including the libraries used within the app. They do not differentiate between the code in the libraries, and the source code

written by the developers. This means that any discoveries by our tools could either come from the app's code, or the included libraries. Previous work has found that higher rated apps have more dependence on libraries [24], so it is quite possible that the discovered issues have more to do with these libraries than the actual code.

Although we offer some potential explanations for our findings, we believe that our work has demonstrated the need for future research in this area.

Does it mean that high-rated apps are less secure?

Possibly. A primary way that over-permissions make an application less secure is that they increase the attack surface [9, 22], 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 [5]. Our findings may only conclude that high-rated apps have a higher rate of *potential* vulnerabilities, and request more permissions in comparison to low-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 source of Android apps, 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 [1]. 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 [9, 23], no static analysis tool is perfect and often inherently contains limitations [5]. Although Stowaway is a powerful static analysis tool which has been used in previous research [15, 23], it does suffer from drawbacks; Stowaway's own authors state that the tool only achieves 85% code coverage [9], 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 [5].

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, 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 [19].

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 work [17] 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.

The permission gap was used as a metric to evaluate potential vulnerabilities in apps. However, it is important to note that over-privileges only represent *possible* app vulnerabilities. Additionally, end users will be very unlikely to detect these as vulnerabilities and will frequently believe that they are needed by the app.

Our study only measured one quality aspect of an app, some of its security metrics. However, users are likely to include numerous factors of an app into their final rating including defects and functionality. Future work may be done to use tools such as JLint[6] or Findbugs[7] to provide a more well robust evaluation of the app.

Apps may also have invocations to the Android API in *dead code*, as it is often the case. This is very common since apps use external libraries that might offer many more functionality than the one the app uses. This means that reported over-privileges may not actually be located in code that is actually accessible by the app. When comparing our results, we only looked at the aggregate results of each tool. For example, we did not break down the results based on the individual metrics used to create the aggregate AndroRisk score and did not examine which specific over-permissions might correlate to lower user reviews.

6. CONCLUSION

Our goal was to determine if low-rated apps suffered from more possible security risks than high-rated apps. We statically analyzed 38,466 Android apps using APKParser for permission usage information, Stowaway for under- and over-permission information, and Androrisk for a generic security risk assessment. According to several static analysis tools, we found that high-rated apps had a greater permissions gap, and a higher Androrisk score. While statistically significant, the correlations are very weak.

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] `http://www.amazon.com/mobile-apps/b?node=2350149011`

[5] `https://f-droid.org/`

[6] `http://jlint.sourceforge.net`

[7] `http://findbugs.sourceforge.net`

[4] A. Bartel, J. Klein, M. Monperrus, and Y. L. Traon. Static analysis for extracting permission checks of a large scale framework: The challenges and solutions for analyzing android. *IEEE Transactions on Software Engineering*, 40(6):617–632, June 2014.

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

[6] 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`, Jan 2016.

[7] 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.

[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] 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.

[10] 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.

[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] 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] 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.

[15] 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.

[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, 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.

[19] 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.

[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] 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.

[24] I. M. Ruiz, M. Nagappan, B. Adams, T. Berger, S. Dienst, and A. Hassan. On the relationship between the number of ad libraries in an android app and its rating. *IEEE Software*, 99(1), 2014.

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

[26] 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.

[27] 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.

[28] 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 & Mobile Devices*, SPSM '14, pages 39–50, New York, NY, USA, 2014. ACM.

[29] X. Wei, L. Gomez, I. Neamtiu, and M. 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.