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

Daniel E. Krutz, Meiyappan Nagappan, and Andrew Meneely
Rochester Institute of Technology, Rochester, NY, USA
{dkrutz, mei, andy}@se.rit.edu

***Abstract*—The success or failure of a mobile application ('app') is largely determined by user ratings. Users expect apps to continually provide new features while maintaining quality, or the ratings drop. However, 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 798 random Android apps in the GooglePlay store with a user rating of less than 3 against 861 apps with a user rating of 3 or greater. From our experiments, we find that several of the security metrics that we collected using static analysis were higher (more risk) in high rated apps compared to low rated apps. For example, while low rated apps request more permissions overall, high rated apps will have more over-permissions and under-permissions than low rated apps. Combining the collected evidence from two static analysis tools, we conclude that, historically, user ratings may be a more all-encompassing measure that does not depend heavily on security (yet).**

I. Introduction

Over the last decade the use of smart mobile devices has exploded. Mobile apps are the software applications that run on mobile devices such as smart phones and tablets. These apps are often distributed to the end users through a centralized online store. There are currently millions of apps in these online stores (eg. Google Play app-store[1] and Apple's App Store[2]) and Android users download more than 1.5 billion applications apps from GooglePlay every month[3]. Apps are a major part of mobile consumer technology and have changed the computing experience of our modern digital society, allowing users to perform a variety of tasks not previously possible in a portable environment.

The success of a mobile app is largely determined by user ratings from a digital storefront ('app store'). Users expect apps to continually provide new features, otherwise poor app store reviews and low ratings can be expected [10]. Developers must frequently update their app's dependencies to keep up with the rapid progress of mobile technology [19]. Most importantly, apps must be secure since they are a crucial entry point into our digital lives.

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 may slip. New security-inspired features may also be perceived by users as cumbersome, leading to lower ratings. Even a vulnerability in a dependency can be detrimental to users, yet developers may not have the resources to thoroughly inspect a third-party framework for security concerns. Experts have even warned [1] 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 that indicates a 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 more than 1600 Android apps from the Google Play app store.

The objective of this study is to investigate the 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 while lower rated applications requested more permissions, and certain security risk metrics were higher in them, most of the security risks were greater in higher rated applications. Based on our empirical evidence, we conclude that user ratings (which captures the user's perception of an app) is an all-encompassing metric that is not yet affected by higher security risks.

The rest of the paper is organized as follows: In Section II 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 III, we present the results of our case study. In Section IV, we discuss the related work. In Section V, we present the threats to validity for our study. And finally in Section VI, we conclude the paper based on our findings.

[1]Google Play app-store: play.google.com/store/apps
[2]Apple's App Store: www.apple.com/iphone/apps-for-iphone
[3]http://developer.android.com/about/

II. Study Design

We first collected a variety of apps from GooglePlay using a modified collection tool and then analyzed them using two well known Android static analysis tools. Our collection and analysis process is shown in Figure 1.

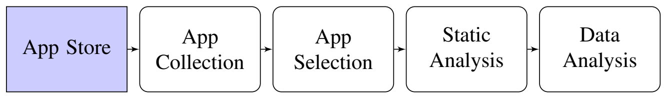

Fig. 1: App Collection and Analysis Process

A. Data Collection

We collected apps from the GooglePlay store with a custom-built collector, which uses *Scrapy*[4] as a foundation. We chose to pull from GooglePlay since it is the most popular source of Android apps[5] and was able to provide other app related information such as the developer, version, genre, user rating, and number of downloads. We downloaded 1,000 apps with a user rating of at least 3, and were able to download 833 apps with a user rating of less than 3. Locating apps with a user rating of less than 3 was much more difficult than finding apps with a user rating of at least 3 [13].

B. Data Selection

We removed all apps with less than 1,000 downloads to limit the effects that rarely used apps would have on our study. This left us with 798 apps with a rating of less than 3, and 861 apps with a user rating of at least 3.

C. Static analysis tools

The next phase was to analyze the apps for potential security risks, and permissions issues. We used two open source static analysis tools in our study: Stowaway [5] and Androrisk[6]. Stowawy evaluates the app for permission gaps, while Androrisk determines the vulnerability risk level. The complete list of metrics (and their definitions) that we collected from Stowaway and Androrisk are in Tables II and III respectively.

We selected Stowaway for determining the permission gap in apps since it is able to state the permissions that were causing it to have over-permissions and under-permissions, while using a static analysis based approach that did not require it to be ran on an Android device or through an emulator. Stowaway has also demonstrated its effectiveness in existing research [5], [14], [17]. Permlyzer [21], a more modern permission detection tool, was not used since its authors have not made it available for download.

We chose Androrisk for several reasons. The first is that the AndroGuard library, which is it a part of, has already been used in a variety of existing research [2], [3], [20]. Androrisk is also a freely available, open source tool which will allow others to replicate our findings. Finally, as a static analysis based vulnerability detection tool, Androrisk was quickly able to effectively determine the risk level of a large number of downloaded apps.

[4]http://scrapy.org

[5]http://www.onepf.org/appstores/

[6]https://code.google.com/p/androguard/

D. Data Analysis

The data collected from the static analysis tools along with user ratings data was analyzed using standard libraries like the *stats* library in R.

III. Results

In this section, we discuss the motivation, approach, and findings for our research question. A more detailed discussion of our results is then provided. Before we present the results to our research questions we present some basic information about the apps used in the study. We have two sets of apps in our case study - 798 apps with a rating lower than 3 stars (low rating apps) and 861 apps with a rating greater than or equal to 3 stars (high rating apps). As shown in Table I, the lower rated apps have a statistically lower number of downloads and size (measured in number of java files). We carried out the following research question with this set of apps.

RQ1: Do apps with lower ratings have more security risks?

Motivation: Android developers operate under a permission-based system where apps must be granted access to various areas of functionality before they may be used. When an Android app is created, developers must explicitly declare in advance which permissions the application will require [5], 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* is thrown. When installing the app, the user is asked to accept or reject these requested permissions. Once installed, the developer cannot remotely modify the permissions without releasing a new version of the app for installation [16], prompting the user if new permissions are required.

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 [5]. This may be due to the lack of granularity of the permission spectrum used by Android, so the developer must often grant more permissions to their app than it actually requires. For example, an application that needs to send information to one site on the internet will need to be given full permissions to the internet, meaning that it may communicate with with all websites [8].

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 [15]. Granting more permissions than the app needs creates security problems since vulnerabilities in the app, or malware, could use these extra

TABLE I: MWU Results for FileSize Settings

Value	Description	Greater or Less	p-value
UserRating	User Rating	L	4.958e-273
Download Count	Number of Downloads	L	3.329e-56
JavaFiles	# of Java Files In App	G	0.005

permissions for malicious reasons. Additionally, this principle limits potential issues due to non-malicious developer errors.

Approach: We use the Stowaway tool to extract the number of permissions used, and the number of over-permissions and under-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. 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.

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. These include the ability to access the internet, manipulate SMS messages or the ability to make a payment. The second is the presence of more dangerous sets of functionality in the app including a shared library, use of cryptographic functions, and the presence of the reflection API.

We apply these two tools to the two sets of apps in our case study - 798 apps with a rating lower than 3 stars (low rating apps) and 861 apps with a rating greater than or equal to 3 stars (high rating apps). We get the distribution for each of the permission and security based metrics from each set of apps. In order to answer our research question we check if the values of permission and risk based metrics are different in high and low rating apps taken as two distinct groups. Our null hypothesis is that there is no difference in the distribution of the various 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.

Findings and Discussion

We present the results of comparing the distributions (hypothesis tests) of the various security risk based metrics from the two static analysis tools Stowaway and Androrisk in Tables II and III respectively. From Table II, we can see that low-rated apps indeed ask for more permissions than high-rated apps. However, the over-permissions and under-permissions are greater in high-rated apps. These results imply that even though low-rated apps request more permissions, they actually use all of them. High-rated apps are asking for permissions that they do not even use, and this is quite dangerous. The high-rated apps also have a larger incidence of under-permissions. This implies that there are features trying to do activities for which they do not have permission for. Such a problem not only indicates reliability issues, but also indicates that developers of the high-rated apps might be trying to get more information than they ask for. Due to the strong permissions setup in Android, such an attempt will fail. However, a failure in Android could result in developers accessing information without even asking the users for permissions.

From Table III, we can see that the overall risk score (FuzzyRisk) in low-rated apps is higher than the high-rated apps. However, when we look at the breakdown into the individual risk metrics, only three of the nine metrics are higher in low-rated apps as compared to high-rated apps: reflection API use (Dex_Reflection), access of internet (Perm_Internet), and access to private data (Perm_Dangerous). The combination of the risk of access to private data and the access to internet, might be the reason why the overall risk score is high even though only three of the nine risk metrics are higher in low-rated apps.

Interestingly, six of the nine risk metrics from Androrisk are lower in low-rated apps as compared to high-rated apps. For example, high-rated apps have a higher risk of manipulating SMS messages as well as loading dex files dynamically. High-rated apps also more likely to ask for the permission to process payments - combined with the risk of loading dex files dynamically, a new executable could possibly take money out of your account as well.

Thus to answer to our research question, low-rated apps do not necessarily have more security risks. While there is some indication that they might have, our data suggests that:

High-rated apps typically have higher security risks

IV. Related Work

There has been a substantial amount of previous research analyzing the effects of permissions on the user's perception of the app. Felt et al. [6] performed two usability studies from users over the internet and in a laboratory setting and found that only 17% of users paid attention to permissions when installing apps. Lin et al. [11] conducted a study analyzing user's comfort levels when apps requested permissions which the user's did not understand why the apps needed them. They found that user's generally felt uncomfortable and may even delete applications when they did not understand why it requested a permission they deemed unnecessary. Egelman et al. [4] 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 allow their apps more permissions to receive

TABLE II: MWU test results of metrics from Stowaway when comparing low- and high-rated apps

Value	Description	Greater or Less	p-value
PermissionCount	# of Permissions in App	G	0.0036
OPrivCount	# Of Overprivileges	L	0.0453
UPrivCount	# Of Underprivileges	L	0.001

TABLE III: MWU test results of metrics from Androrisk when comparing low- and high-rated apps

Value	Description	Greater or Less	p-value
Dex_Native	Presence of loading a shared library	L	4.571e-13
Dex_Dynamic	Presence of loading dynamically a new dex file	L	2.719e-20
Dex_Crypto	Presence of crypto functions	L	2.123e-05
Dex_Reflection	Presence of the reflection API	G	0.0281
Perm_Money	Presence of permissions which can result to a payment	L	0.0009
Perm_Internet	Presence of permissions which can access to internet	G	1.205e-18
Perm_SMS	Presence of permissions which can manipulate sms	L	0.0008
Perm_Dangerous	A higher-risk permission that would give a requesting application access to private user data or control over the device that can negatively impact the user	G	6.139e-13
Perm_Signature	A permission that the system grants only if the requesting application is signed with the same certificate as the application that declared the permission	L	0.037
FuzzyRisk	Overall Risk Score of App	G	1.790e-14

targeted advertisements if it would save them .99 cents on the purchase of the app. Stevens et al. [18] found that certain permissions that are not popular among developers were frequently misused, possibly due to a lack of documentation. Similar to the above papers, we examine the permissions and security risks in Android apps. However, we compare them against user ratings, which are user's perspective of an app that was obtained without us soliciting it from them.

App ratings have demonstrated their importance in other areas of research as well. Harman et al. [7] found a strong correlation between the rating and the number of app downloads. Linares-Vasquez et al. [12] found that change and fault-proneness of the APIs used by the apps negatively impacts their user ratings. Khalid et al. [9] examined 10,000 apps using FindBugs and found that warnings such as 'Bad Practice', 'Internationalization', and 'Performance'categories are typically found in lower rated apps. Their primary discovery was 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, we look at permission and security risks unlike earlier works.

V. Threats to Validity

While GooglePlay is the largest Android market place, it is not the only source for Android apps. Alternatives include the Amazon app store, GetJar, F-Droid and a multitude of other sources. Other studies may choose to include apps from these other sources. Additionally, we chose apps at random and only selected a total of 1,659 apps, which is a small minority of the over 1.5 total Android apps available[7]. Given that this is a random sample, however, we believe that it is representative of the Android application population.

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

Although Stowaway and Androrisk have been used in a substantial amount of previous research, they are by no means perfect or are the only over-permission and under-permission, and risk assessment tools available. While no tools are likely perfect, Stowaway has only been found to achieve approximately 85% code coverage of the Android API [5].

While user ratings have been substantially analyzed in previous research, popular apps will receive ratings from only 2-3% of its users [22]. This means that only a small subset of users are responsible for the overall ratings an app receives and may not be representative of what all users believe about the app. However, mobile application developers still need their ratings to be high to be successful.

VI. Conclusion

Summary: We statically analyzed more than 1,600 Android apps (both high-rating and low-rating apps) for security threats and found that high-rated apps typically had lower security based issues but higher permission based issues as compared to low-rated apps.

Recommendations: Even though high-rated apps have greater risks in most cases, the use of static analysis tools is feasible for understanding security related issues. Therefore, we recommend that app developers run these static analysis tools on their apps before releasing them as they have an opportunity to correct issues before end users download, use and rate their apps.

References

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

[2] 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).

[3] 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 &#38; Communications Security*, CCS '13, pages 73–84, New York, NY, USA, 2013. ACM.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] I. J. Mojica Ruiz. Large-scale empirical studies of mobile apps. 2013.

[14] 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. ACM.

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

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

[17] 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.

[18] 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, May 2013.

[19] 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. IBM Corp.

[20] 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.

[21] W. Xu, F. Zhang, and S. Zhu. Permlyzer: Analyzing permission usage in android applications. In *Software Reliability Engineering (ISSRE), 2013 IEEE 24th International Symposium on*, pages 400–410, 2013.

[22] B. Yan and G. Chen. Appjoy: Personalized mobile application discovery. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, MobiSys '11, pages 113–126, New York, NY, USA, 2011. ACM.