Short Paper: A Longitudinal Study of Financial Apps in the Google Play Store

Vincent F. Taylor and Ivan Martinovic

Department of Computer Science,
University of Oxford,
Oxford, United Kingdom.
{vincent.taylor,ivan.martinovic}@cs.ox.ac.uk

Abstract. Apps in the `FINANCE` category constitute approximately 2% of the 2,000,000 apps in the Google Play Store. These apps handle extremely sensitive data, such as online banking credentials, budgets, salaries, investments and the like. Although apps are automatically vetted for malicious activity before being admitted to the Google Play Store, it remains unclear whether app developers themselves check their apps for vulnerabilities before submitting them to be published. Additionally, it is not known how financial apps compare to other apps in terms of dangerous permission usage or how they evolve as they are updated. We analyse 10,400 apps to understand how apps in general and financial apps in particular have evolved over the past two years in terms of dangerous permission usage and the vulnerabilities they contain. Worryingly, we discover that both financial and non-financial apps are getting more vulnerable over time. Moreover, we discover that while financial apps tend to have less vulnerabilities, the rate of increase in vulnerabilities in financial apps is three times as much as that of other apps.

1 Introduction

Android is the dominant mobile operating system with control of 84.7% of the smartphone market as of 2015 Q3, dwarfing its nearest rival, iOS, at 13.1% [9]. Smartphone users use over 26 different apps per month, and spend more than one hour per day using apps on average [12]. In the United Kingdom, banking using a mobile device such as a smartphone or tablet has already overtaken the act of going into a branch or using a PC to bank [4]. Recently, Finance Monthly reported that usage of finance and banking apps rose 17% among "affluent middle class" customers. Along similar lines, Google reports that 75% of users use only one or two finance apps, but that 44% of users use these finance apps on a daily basis.

Fraudsters and other adversaries have long been known to exploit victims for the greatest financial gain, and with the rising popularity of financial apps, we expect their attention to turn there. Previous work has analysed apps in the Google Play Store as a whole [6], but it remains unclear whether a one-size-fits-all approach to understanding smartphone apps in general properly encapsulates

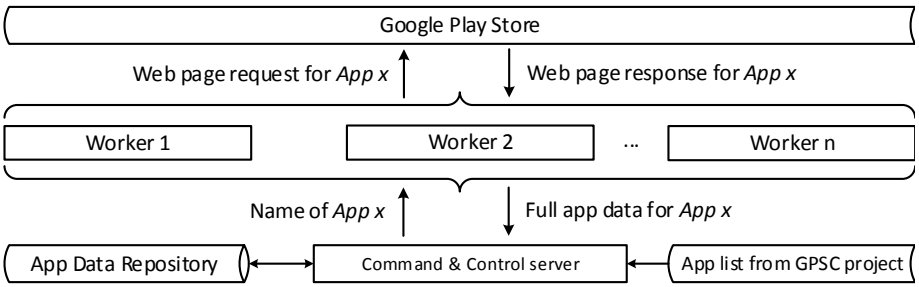

Fig. 1. Highly-scalable cloud-based crawler architecture.

the idiosyncrasies of financial apps in particular. Indeed, financial apps handle more sensitive information than most typical apps and thus have a requirement for the secure storage, processing and transmission of this data.

To address this gap in the literature, we performed several tasks. We collected snapshots of the entire Google Play Store quarterly over a two-year period to understand how apps in general and financial[1] apps in particular have evolved in terms of dangerous[2] permission usage. Additionally, we used our most recent snapshot of apps to compare and contrast financial apps to the remainder of apps in the Google Play Store. Finally, we used open-source Android app vulnerability scanning tools to understand how financial apps compare to other apps in terms of the vulnerabilities they contain and how this changes as apps are updated by their developers.

Contributions. Specifically, our contributions are as follows:

- We analyse how financial apps have evolved over the past two years when compared to other apps in terms of their dangerous permission usage.
- We perform security analyses on 10,400 apps using static vulnerability analysis tools to understand how financial apps compare to other apps in terms of the vulnerabilities they contain and how they change as apps are updated.

Roadmap. Section 2 overviews the evolution of dangerous permission usage; Section 3 describes how our dataset was collected and the vulnerabilities examined; Section 4 presents our vulnerability scanning results; Section 5 discusses our observations and future work; Section 6 surveys the most related work; and finally Section 7 concludes the paper.

[1] We consider financial apps to be those apps listed in the Google Play Store under the `FINANCE` category.

[2] Dangerous permissions guard access to sensitive user data and must be requested by apps and approved by users before the relevant data can be accessed [3].

Table 1. Mean dangerous permission usage (and percentage change) across apps over the two-year period based on number of app downloads.

Downloads	ALL apps			FINANCE apps		
	OCT-2014	SEP-2016	Change	OCT-2014	SEP-2016	Change
1-1K	3.13	3.16	+0.96%	2.75	2.85	+3.64%
1K-1M	2.37	2.45	+3.38%	3.20	3.43	+7.19%
1M-5B	3.40	3.58	+5.29%	5.62	6.44	+14.59%

2 Google Play Store Analysis

Our first task was to understand how financial apps have evolved in terms of dangerous permission usage. To capture app metadata, we developed a highly-scalable cloud-based crawler as shown in Fig. 1. This crawler is run quarterly and is capable of harvesting full app metadata in less than 48 hours. Our crawler is informed of all the apps in the Google Play Store by the Google Play Store Crawler Project (GPSC) [11]. Using our crawler, we obtained approximately two years of app metadata[3], from OCT-2014 to SEP-2016, on all available apps.

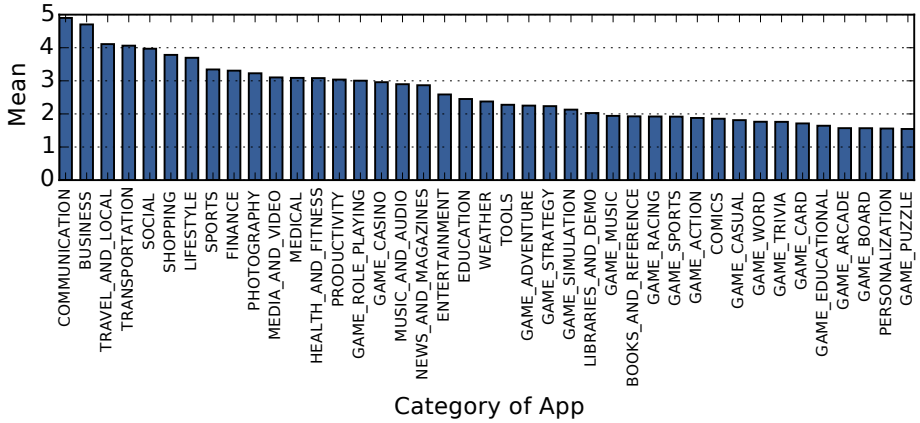

Fig. 2. Mean number of dangerous permissions used per category of app.

Fig. 2 shows how the number of dangerous permissions per category of app varied. Apps in the FINANCE category use among the highest number of dangerous permissions at 3.3. Moreover, as shown in Table 1, financial apps had a greater percentage increase in the number of dangerous permissions used over the last two years when compared to all apps. From Fig. 3, we can see that financial apps use permissions typical to that of other apps, except permissions used to access a user's location, camera, and contacts, as notable examples.

[3] Our app metadata is available to the research community upon request.

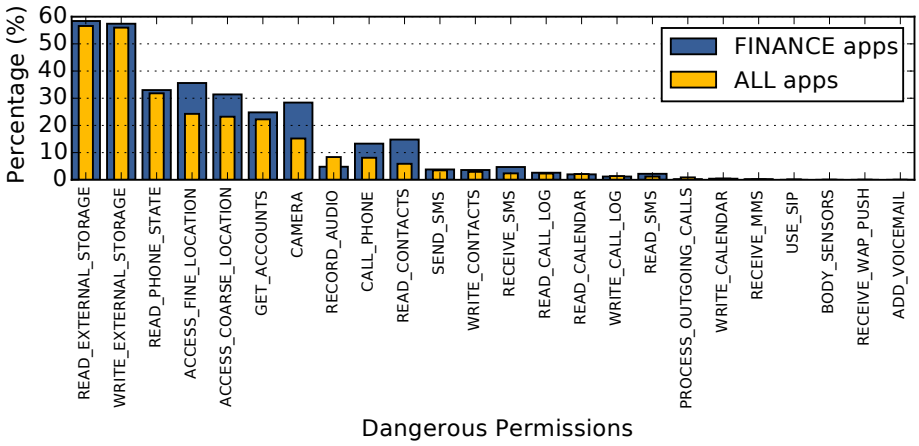

Fig. 3. Permission usage in FINANCE apps compared to ALL apps.

3 Dataset and Tools

The next step was to understand how the vulnerabilities contained within financial apps changed as apps were updated. To construct our app dataset, we randomly chose and downloaded 200 apps in the `FINANCE` category and 5,000 apps from all `OTHER` categories. Additionally, we leveraged the PlayDrone [14] dataset to get the corresponding `.apk` files for these apps from two years ago for a total of 10,400 apps. Our dataset is summarised in Table 2.

Table 2. Dataset of apps used in the analysis.

Category	Dataset Name	# of APKs	Source	Date
`FINANCE`	`FIN-OLD`	200	PlayDrone	Oct-2014
	`FIN-NEW`	200	Google Play	Sep-2016
`OTHER`	`OTH-OLD`	5,000	PlayDrone	Oct-2014
	`OTH-NEW`	5,000	Google Play	Sep-2016

3.1 Vulnerabilities Analysed

The vulnerabilities that were analysed are listed in Table 3. Vulnerabilities were synthesised from the OWASP Top 10 [13] which lists common vulnerabilities affecting mobile apps. We used two popular app security analysis frameworks to analyse apps for vulnerabilities: AndroBugs [1] and MobSF [2]. These frameworks leverage static code analysis and are lightweight and scalable, making them suitable for our purpose.

Static analysis tools suffer from their reduced ability to handle dynamic programming features such as reflection and dynamic code loading. Thus our vulnerability scanning may fail to detect issues that only emerge at runtime. For this reason, the number of vulnerabilities reported should be considered a lower bound on the actual number of vulnerabilities present within apps.

Table 3. List of vulnerabilities considered.

Identifier	Description	Tool Used
INF-DISC-WRLRD	App uses world readable/writeable files	AndroBugs
INF-DISC-PRVDR	ContentProvider exported but not secured	
INF-DISC-KSNPW	Keystores not protected by a password	
SSL-TLSX-PLAIN	Sending data over plain HTTP	AndroBugs
SSL-TLSX-INVLD	Invalid SSL certificate verification	
SSL-TLSX-WVIEW	Improper WebView certificate validation	
BRK-CRYP-ECBMD	Use of the ECB cryptographic function	MobSF
BRK-CRYP-RANDG	Use of insecure random number generators	
OTH-MISC-INTNT	Starting services with implicit Intents	AndroBugs
OTH-MISC-DEBUG	App is debuggable	
BIN-ROOT-DTECT	App does not have root detection	MobSF

4 Results

The results of our vulnerability analysis is shown in Table 4. Worryingly, both classes of apps became more vulnerable as they were updated for a majority of the vulnerabilities considered. For financial apps however, the prevalence of vulnerabilities overall was lower when compared to other apps. While this is welcome, we note that the average percentage increase in vulnerabilities in financial apps was approximately three times that of other apps.

Non-financial apps had four types of vulnerabilities that actually improved as apps were updated: SSL-TLSX-VERIF, SSL-TLSX-WVIEW, OTH-MISC-DEBUG and BIN-ROOT-DTECT. The only vulnerability that improved for financial apps was BIN-ROOT-DTECT. The Top 4 vulnerabilities that had the highest increase in prevalence were shared between financial and non-financial apps. These vulnerabilities involved apps creating world readable/writeable files (INF-DISC-WRLRD), using unsecured ContentProviders (INF-DISC-PRVDR), generating random numbers insecurely (BRK-CRYP-RANDG) and using implicit intents to start services (OTH-MISC-INTNT). Unsecured ContentProviders and world-readable files introduce the possibility of malicious apps on a device reading data stored by a vulnerable app. Considering that financial apps handle sensitive data, care should be taken by app developers to ensure that such data is stored securely on the device.

Table 4. Percentage of apps within each dataset containing one or more of each studied vulnerability.

Vulnerability	OTH-OLD (%)	OTH-NEW (%)	FIN-OLD (%)	FIN-NEW (%)
INF-DISC-WRLRD	16.5	24.7	10.5	20.5
INF-DISC-PRVDR	2.22	2.92	2.00	3.00
INF-DISC-KSNPW	2.32	2.34	2.00	2.00
SSL-TLSX-PLAIN	80.1	80.7	75.5	77.0
SSL-TLSX-VERIF	15.4	14.6	15.5	16.0
SSL-TLSX-WVIEW	9.74	9.21	9.50	10.0
BRK-CRYP-ECBMD	12.7	12.7	10.5	11.1
BRK-CRYP-RANDG	59.3	63.8	54.5	61.1
OTH-MISC-INTNT	3.22	5.19	2.50	7.50
OTH-MISC-DEBUG	2.30	2.06	2.00	2.00
BIN-ROOT-DTECT*	95.2	93.4	96.5	92.9

*Root detection may be implemented in many ways, thus false positives may be present in our result, and consequently we consider these numbers an upper bound.

5 Discussion

It is welcome to observe that financial apps on average have a lower prevalence of vulnerabilities. However, it is worrying that these numbers are increasing, and indeed increasing faster than that of other apps. Given that financial apps potentially handle very sensitive information, great care needs to be taken by app developers to safeguard the data that their apps use.

As a first step, app developers should familiarise themselves with the OWASP Top 10 [13] to understand the typical security problems that affect mobile apps. By gaining a better understanding of typical security problems, app developers can avoid common mistakes that make their apps easily exploitable by adversaries. App developers can also leverage any of the myriad open-source static/dynamic vulnerability analysis tools to check their apps for vulnerabilities before publishing them to app stores. In some cases, app developers may not be the primary source of the vulnerabilities contained within their apps. Many app developers unwittingly use vulnerable libraries and introduce vulnerabilities into otherwise secure apps. App developers must take care to ensure that they always use up-to-date versions of libraries whenever they update their apps.

The official Android app store, Google Play, can be a catalyst for improving the quality of apps by performing vulnerability analysis checks on apps at the time when they are submitted to be published. During this research, we observed that scanning apps for vulnerabilities takes less than one minute on average. Publishing an app to the Google Play Store already takes up to several hours, so we expect that lightweight vulnerability scanning will not cause a noticeable delay. Apps containing vulnerabilities can be returned to app developers for

fixing, penalised in search results, or flagged as being vulnerable when presented to users.

Static analysis alone does not paint the full picture of what is happening inside apps. For future work, we plan to use dynamic analysis tools to further understand the vulnerabilities contained within apps, as well as explore a wider range of app vulnerabilities.

6 Related Work

Viennot et al. [14] developed a tool called PlayDrone and used it to perform the first indexing of apps in the Google Play Store. We leverage their dataset to obtain old versions of apps to perform our longitudinal analysis of vulnerability evolution. Along similar lines, Book et al. [5] perform a longitudinal analysis of permission usage in ad libraries. The authors discover that not only have ad libraries gained greater sensitive access to devices over time, but they typically get access that risks user privacy and security. We complement this analysis by evaluating how permissions have increased in apps overall and within financial apps specifically.

A number of authors identified different classes of vulnerabilities in Android apps and proposed various tools to detect them. We list a few for brevity. Fahl et al. [8] investigated SSL/TLS related vulnerabilities using a tool called Mallodroid. The authors found that approximately 8% of the apps that were examined were potentially vulnerable to man-in-the-middle attacks. Equally important, Lu et al. [10] investigated Android apps being vulnerable to component hijacking attacks. Subsequently, Egele et al. [7] investigated whether apps were using cryptographic APIs securely. They found that 88% of the apps investigated made at least one mistake when using cryptographic APIs. Complementary to these pieces of work, we use static analysis tools to evaluate the extent to which Android apps currently suffer from these and other vulnerabilities. Additionally, we examine how the prevalence of vulnerabilities has changed as apps have been updated, as well as how financial apps compare to regular apps as it relates to vulnerability evolution.

7 Conclusion

In this paper, we investigated dangerous permission usage in Android apps in general and in finance apps in particular. We discovered that while both classes of apps had increases in the number of dangerous permissions used, financial apps typically used more dangerous permissions and also had greater percentage increases in permission usage. Additionally, financial apps tended to use location, camera and contacts permissions more than other apps. By doing vulnerability analysis of apps, we observed that apps tend to become more vulnerable as they are updated. While financial apps were less likely to contain vulnerabilities overall, as they were updated their prevalence in containing vulnerabilities increased three times as much as other apps. As users become more comfortable with using

smartphone apps for sensitive tasks, it becomes imperative that app developers take appropriate measures to ensure that sensitive data remains safe, whether it is stored on a device or transmitted over a network.

Acknowledgement. Vincent F. Taylor is supported by a Rhodes Scholarship and the UK EPSRC.

References

1. AndroBugs Framework. https://github.com/AndroBugs/AndroBugs_Framework.
2. Mobile Security Framework. https://github.com/ajinabraham/Mobile-Security-Framework-MobSF.
3. Requesting Permissions. https://developer.android.com/guide/topics/permissions/requesting.html.
4. BBA. Mobile phone apps become the UKs number one way to bank. https://www.bba.org.uk/news/press-releases/mobile-phone-apps-become-the-uks-number-one-way-to-bank/, June 2015.
5. T. Book, A. Pridgen, and D. S. Wallach. Longitudinal analysis of Android ad library permissions. *arXiv preprint arXiv:1303.0857*, 2013.
6. B. Carbunar and R. Potharaju. A Longitudinal Study of the Google App Market. In *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, ASONAM '15, pages 242–249, New York, NY, USA, 2015. ACM.
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 and Communications Security*, CCS '13, pages 73–84, New York, NY, USA, 2013. ACM.
8. S. Fahl, M. Harbach, T. Muders, L. Baumgärtner, B. Freisleben, and M. Smith. Why Eve and Mallory Love Android: An Analysis of Android SSL (in)Security. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 50–61, New York, NY, USA, 2012. ACM.
9. Gartner. Gartner Says Emerging Markets Drove Worldwide Smartphone Sales to 15.5 Percent Growth in Third Quarter of 2015. http://www.gartner.com/newsroom/id/3169417, November 2015.
10. L. Lu, Z. Li, Z. Wu, W. Lee, and G. Jiang. CHEX: Statically Vetting Android Apps for Component Hijacking Vulnerabilities. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, pages 229–240, New York, NY, USA, 2012. ACM.
11. Marcello Lins. Google Play Apps Crawler. https://github.com/MarcelloLins/GooglePlayAppsCrawler.
12. Nielson. Smartphones: So Many Apps, So Much Time. http://www.nielsen.com/us/en/insights/news/2014/smartphones-so-many-apps–so-much-time.html, July 2014.
13. OWASP. Projects/OWASP Mobile Security Project - Top Ten Mobile Risks. https://www.owasp.org/index.php/Projects/OWASP_Mobile_Security_Project_-_Top_Ten_Mobile_Risks.
14. N. Viennot, E. Garcia, and J. Nieh. A Measurement Study of Google Play. In *The 2014 ACM International Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '14, pages 221–233, New York, NY, USA, 2014. ACM.