AndroZooOpen: Collecting Large-scale Open Source Android Apps for the Research Community

Pei Liu, Li Li, Yanjie Zhao, Xiaoyu Sun, John Grundy
Faculty of Information Technology, Monash University, Australia
{pei.liu,li.li,yanjie.zhao,xiaoyu.sun,john.grundy}monash.edu

ABSTRACT

It is critical for research to have an open, well-curated, representative set of apps for analysis. We present a collection of open-source Android apps collected from several sources, including Github. Our dataset, AndroZooOpen, currently contains over 45,000 app artefacts, a representative picture of Github-hosted Android apps. For apps released on Google Play, metadata including categories, ratings and user reviews, are also stored. We share this new dataset as part of our ongoing research to better support and enable new research topics involving Android app artefact analysis, and as a supplement dataset for AndroZoo, a well-known app collection of close-sourced Android apps.

CCS CONCEPTS

• **Software and its engineering → Software libraries and repositories**.

KEYWORDS

Android, Open-source, AndroZoo, AndroZooOpen

ACM Reference Format:
Pei Liu, Li Li, Yanjie Zhao, Xiaoyu Sun, John Grundy. 2020. AndroZooOpen: Collecting Large-scale Open Source Android Apps for the Research Community. In *17th International Conference on Mining Software Repositories (MSR '20), October 5–6, 2020, Seoul, Republic of Korea.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3379597.3387503

1 INTRODUCTION

Android is now the most popular mobile operating system with over 80% market share [5]. Due to the increase in affordability and adoption of smart touchscreen and powerful handheld devices such as smartphones, tablets, and smartwatches, etc., mobile application development has witnessed an unprecedented growth in recent years. In particular, the millions of Android Apps produced by developers have empowered over a billion users' daily tasks, ranging from emails and games to daily health monitoring.

Due to this popularity and the openness of the Android platform, researchers and practitioners have spent a large amount of effort in improving the quality and security of Android apps [19, 24, 30]. For example, Li et al. have identified more than 100 papers leveraging static analysis techniques to analyze Android apps, and around 60 articles targeting the repackaging problem of Android apps [23, 24]. Martin et al. have found over 180 works proposed to support the analysis of mobile apps through App Store mining [30].

A key reason for this activity and success of analyzing Android apps is the development by the community of many open datasets for supporting the evaluation of these approaches. For example, Zhou et al. [37] introduce the Genome project, which contains thousands of malicious Android apps (also known as malware) and has now been leveraged by hundreds of research works. Similarly, Arp et al. [8] released the Drebin dataset, a larger dataset of malicious Android apps, that has also been well leveraged. Regarding benign Android apps (regular apps that are available on popular app markets and are reported by VirusTotal as such), the most representative dataset is currently AndroZoo [7, 27], which was mainly collected by researchers at the University of Luxembourg. They have made AndroZoo available to other researchers online. To date, this dataset contains more than 10 million Android apps, collected from various app markets, including the official Google Play store.

Unfortunately, all the aforementioned datasets are mainly made up of *close-sourced Android apps*. This is to be expected, as Android apps are normally only released without source artifacts. However, in many cases, the source code of Android apps is also needed to support a detailed evaluation [12, 25]. For example, among 24 papers targeting Android published in the research track of the Mining Software Repositories (MSR) conference (2015-2019), only five of them have leveraged open-source apps (cf. Scalabrino et al. [34], Habchi et al. [16], Nayebi et al. [32], Ahmad et al. [6], Bao et al. [9]). The numbers of open-source apps used however are all less than 2,000, despite that both F-Droid and Github being used to discover apps. Considering that there are over 45,000 open-source Android apps available, many of these have been overlooked. Subsequently, the experiments conducted in these research efforts may not be generalizable or representative of all the currently available open-source Android apps.

We hypothesize that one of the key reasons causing only a small number of open-source Android apps (to date, concerned by the MSR community) is that our community lacks a well-curated, representative public dataset of open-source Android apps. The fact that all five of these MSR articles have leveraged F-Droid to find apps confirms this hypothesis. Indeed, to the best of our knowledge, F-Droid is one of the few repositories that manage open-source Android apps, although the number of managed apps is quite small, i.e., roughly 2,000. Another dataset introduced by Krutz et al. [20] contains an even smaller number of apps, i.e., 1,179 apps. To fill this gap, we provide to the community a growing dataset of open-source Android apps, namely *AndroZooOpen*. At the moment, *AndroZooOpen*

MSR '20, October 5–6, 2020, Seoul, Republic of Korea

ACM ISBN 978-1-4503-7517-7/20/05…$15.00
https://doi.org/10.1145/3379597.3387503

contains 46,521 app repositories providing a representative picture of open-source Android apps available on Github and F-Droid. Whenever possible, in addition to the metadata of the repositories, we also retrieve the Google Play pages (if published on Google Play) and their AndroZoo app artifacts (if available in AndroZoo) for these apps.

2 DATA COLLECTION

We describe the data sources leveraged to build *AndroZooOpen*. We also discuss the key challenges we faced when crawling and retreiving data from these data sources.

2.1 Data Source

F-Droid and Github are currently the most popular data sources leveraged by researchers to collect open-source Android apps. In this work, we use these two data sources to collect open-source Android apps. Additionally, for open-source apps released on Google Play, we supplement this data with their metadata listed on Google Play, including their user ratings and reviews.

Github is a subsidiary of Microsoft that offers online source code management and distributed software development version control using Git. With over 100 million software repositories, Github is currently the most significant code hosting site in the world and the most preferred source to conduct research related to mining software repositories.

F-Droid is an online repository dedicated to maintaining a collection of free and open-source Android apps, including their source artefacts and code. These are often hosted on popular code hosting sites such as Github and Gitlab. Currently, F-Droid contains 2,078 apps covering different topics ranging from *Utility Tools* to *Games*.

Google Play is the official app store for the Android operating system that allows developers to publish and manage their apps developed with the Android Software Development Kit (SDK). Its users can browse and download apps to meet their requirements. Google Play also provides various metadata information for users to better understand them. Representative metadata information includes app description, app overall user rating, user reviews, etc. These metadata have been demonstrated to be very useful for improving the quality and security of the apps, and hence have been frequently leveraged by researchers and practitioners to complement their approaches.

AndroZoo is a growing collection of closed-source Android apps. We leverage this dataset to harvest open sourced app artifacts and app lineage (i.e., the historical versions of the same app).

2.2 Collecting Challenges

We now enumerate several challenges keeping us from immediately collecting open-source Android apps and share our experiences in resolving each of them, to enable other researchers and practitioners to replicate our work.

First, as mentioned earlier, there are over 100 million software repositories hosted on Github. **How can we quickly and accurately locate open source Android app repositories from them?** It is not practically possible to go through all of them. Fortunately, Github provides the concept of *topics* to categorize repositories. At present Github provides over 180 topics, among which "Android" is one of them offered for Android-related repositories. In this work, we rely on this topic to locate Android app related repositories.

Star	Year 2018	Year 2019
1	800	1300
2	300	200
3	200	100
4	100	50

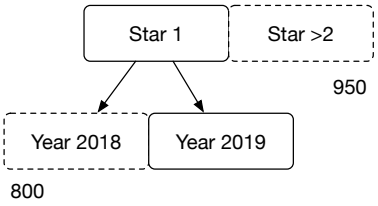


Figure 1: An illustrative example of divide-and-conquer strategy adopted in our approach.

Second, given a repository with android as its topic, **it does not necessarily mean the repository contains source Android app artefacts**. Indeed, this repository can be Android-related libraries, architecture designs, books, etc. For example, one of the most popular repositories, namely *justjavac/free-programming-books-zh_CN*, received 62.9K stars, is not an Android app repository, although android is one of its topics. To this end, we further clone these repositories (i.e., having android as one of their topics) and use analysis scripts to check if the file structure is that of a source Android app, e.g., containing *AndroidManifest.xml* configuration files and main launcher Activity java file, etc., which are essential to all the Android apps. This process is nonetheless time-consuming.

Third, there are **certain limitations enforced by Github to avoid potential Distributed Denial-of-Service (DDoS) attacks**. These make it further challenging for locating Android app repositories. To begin with, for each search request, Github will only return up to 100 results and up to 1000 results via pagination. However, the number of Android-related repositories is over 62,000[2]. Therefore it is not possible to crawl the full list of Android-related repositories. To cope with this, we propose a divide-and-conquer approach, for which we attempt to divide the original problem (i.e., to crawl all the Android-related repositories) to smaller problems (e.g., to crawl all the Android-related repositories that receive 2 stars and are created in 2018) and resolve them respectively. Figure 1 illustrates a simplified example demonstrating the working process of this divide-and-conquer strategy. Since we cannot crawl all the repositories at once – there are 3,050 repositories as shown in the table on the left – we first divide the problem into two small problems: (1) crawling repositories with one star and (2) crawling repositories with at least two stars. Unfortunately, the first small problem still results in more than 1,000 items. Hence, we need to divide this problem further. We further leverage the creation time of Github repositories to divide the problem so as to reduce the search scope for each request.

Last but not least, once Android app repositories are identified, **we need a robust way to collect all of their relevant metadata**. This includes the ones provided by Github (such as the creation date of the repository, the date of the last commit, the total number of commits, etc.), and by Google Play (such as app description, user rating, user reviews, etc.). For Github, we leverage the RESTful APIs provided by Github to achieve the purpose. However, these APIs are limited by Github in terms of their calling frequencies: Requests using Basic Authentication or OAuth is limited to 5,000 per hour while unauthenticated requests are limited to only 60 per hour[4].

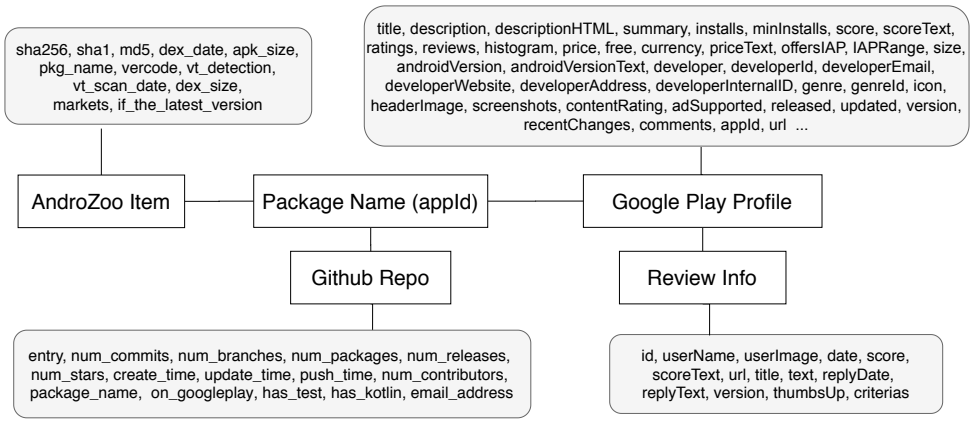


Figure 2: The simplified schema of *AndroZooOpen*.

To overcome these limitations, we have to slow down the crawling process.

While crawling the repository metadata from Github, we notice that the number of contributors provided by Github is limited to 500. If a given repository has more than 500 contributors, the RESTful API provided by Github will only return the first 500 contributors and regard all the remaining ones as anonymous contributors. Therefore, the metadata of repository contributors we collected in this work may not be the same as the one showing on the website.

In order to collect any app metadata from Google Play, we resort to the AndroidManifest.xml to extract the unique package name (also known as appId) to locate the app pages on Google Play (app metadata is then extracted from the located web pages). **This extraction is however not always straightforward**. Some apps declare their package name via Macro (e.g., <%= appPackage %>), where the actual package names are defined at different locations. Additional steps are hence implemented to extract the unique package name in such app repositories.

3 *ANDROZOOOPEN*

Our dataset currently contains 46,521 open-source Android app repositories, which, to the best of our knowledge, is the first and by far the largest dataset ready to be accessed by other researchers and practitioners. At the end of this section, we give some example statistics to help readers better understand the composition of our dataset.

3.1 Accessing the Dataset

We provide an HTTP API for users to download our dataset as a single compressed artifact (i.e., a zip file) [1]. As highlighted in Figure 2, this artifact contains all the metadata we have collected for *AndroZooOpen*, including Github metadata, apps' Google Play profile and user reviews, as well as other the relevant metadata (including the APKs) gathered in AndroZoo.

We further transform the *AndroZooOpen* dataset into a knowledge graph[1] and upon which we offer an online service allowing users to search open-source Android app repositories with detailed search terms [31, 36].

[1]Knowledge Graph (KG) announced by Google in May 2012 was used to integrate its search engine to enhance the search result with information from a variety of sources.

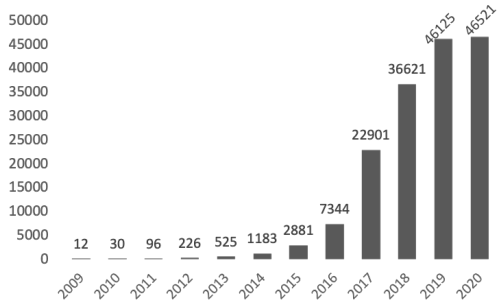


Figure 3: The distribution of open-source Android apps over creation time.

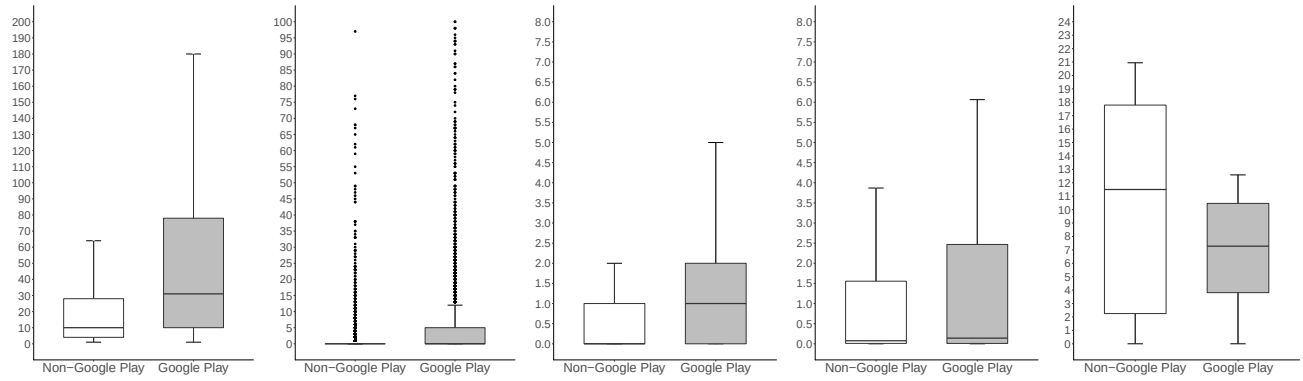
3.2 Example Statistics

Figure 3 illustrates the distribution of apps according to their creation time. Open-source Android apps have been uploaded to Github from as early as 2009, just after Android OS was introduced. Since then, the number of app repositories continuously increases with several leaps in recent years (e.g., there are 15,557 new app repositories created on Github in 2017).

Table 1 further highlights the composition of our dataset. Among the 46,521 collected repositories, as shown in the second column, we were able to locate 3,316 of them on Google Play, accounting for roughly only 7% of total apps. This evidence indicates that real-world Android apps are usually not open-sourced. Among the 3,316 repositories that have their apps released on Google Play, 2892 of them have their released app versions included in AndroZoo. These apps, including their lineages, are also included in our dataset. F-Droid hosts 2,078 open-source Android apps that are also included in our dataset. Among the 2,078 F-Droid apps, 1,646 of them have their source code maintained on Github, as shown in Table 1.

Table 1: Breakdown of *AndroZooOpen*.

Total	Google Play (AndroZoo)	F-Droid (Github)	Kotlin	Test Case
46,521	3,316 (2892)	2,078 (1,646)	10,202	34,174



(a) # Commits. (b) # Releases. (c) # Stars. (d) Active Time. (e) Average Update Interval.

Figure 4: Comparison results between app repositories that have their apps uploaded to Google Play and those that are not.

In May 2017, Kotlin was announced as an officially supported language for Android Platform development by Google. We made effort to check the adoption of Kotlin in our dataset. As shown in Table 1, 10,202 apps are written in Kotlin, accounting for roughly 22% of the total app repositories. If we only consider such repositories created after May 2017, the ratio of Kotlin written apps increases to 30.19%. To support automated testing of Android apps, we further look into the source code of open-source Android app repositories to check if test cases are provided by the app developers. Our empirical investigation reveals that over 70% of the collected app repositories have provided test cases, which could be leveraged to check code correctness against evolutionary changes.

4 LEVERAGING *ANDROZOOOPEN*

As an example demonstrating the usefulness of our dataset, we leverage our *AndroZooOpen* dataset to conduct a lightweight comparative study of open-source Google Play and non-Google Play apps. Our hypothesis is that Google Play apps should be more mature (i.e., bigger code size, more frequently updated) than such open-source apps that are not released on Google Play. To evaluate this hypothesis, we resort to our dataset to form a control group to evaluate this hypothesis empirically. The control group contains (1) 3,316 app repositories that have their apps uploaded to Google Play, and (2) the same number of app repositories that are randomly selected from the remaining repositories.

Based on the control group, we empirically compare the two sets of open-source app repositories in terms of their size (i.e., number of commits, number of releases, number of stars) and their updatability (i.e., active time (last update time - create time) and average update interval (active time divide number of commits). Figure 4 highlights the comparison results. Regarding size, app repositories that have their apps uploaded to Google Play indeed have more commits, more releases, as shown in Figure 4(a) and Figure 4(b), respectively. This evidence confirms our hypothesis that Google Plays are generally more mature than non-Google Play apps. This evidence is further backed up by the fact that Google Play app repositories receive more developer stars than that of non-Google Play app repositories (cf. Figure 4(c)). Moreover, as illustrated in Figure 4(d) and Figure 4(e), Google Play app repositories generally have longer active time and are updated more frequently than that of non-Google Play repositories. This significant result (as proofed by Mann-Whitney-Wilcoxon (MWW) tests) once again ramparts the correctness of our hypothesis.

The aforementioned study is just an example demonstrating the usefulness of our dataset. We believe our dataset could be leveraged to support many other research studies, including but not limited to study the code smells of Android apps [17, 33], the fixes of bugs [18], energy anti-patterns and performance bottlenecks [11, 12], security vulnerabilities [13, 22, 28, 29], and compatibility issues [10, 25, 26, 35], the evolution of open-source Android apps [14, 15, 21], and so on.

5 LIMITATION

The Github repositories collected are all under the topic of Android [3]. Unfortunately, there are likely other open-source Android app repositories that do not come with the Android topic and thereby overlooked by our approach. Furthermore, open-source Android apps may not only be hosted on Github, but also hosted on other online code repositories, such as Bitbucket or Gitlab, which have not been taken into account at the moment. We plan to consider them in our future work as part of our endeavor towards offering the community a continuously growing large collection of open-source Android apps.

6 CONCLUSION

In this work, we presented to the research community the *AndroZooOpen* dataset containing over 45 thousand of open-source Android apps and their metadata and reviews from Google Play, as well as their connections with the most prominent Android App repository AndroZoo.

ACKNOWLEDGMENT

This work was supported by the Australian Research Council (ARC) by a Laureate Fellowship project FL190100035, a Discovery Early Career Researcher Award (DECRA) project DE200100016, and a Discovery project DP200100020.

REFERENCES

[1] 2020. *AndroZooOpen.* http://knowledgezoo.xyz/AndroZooOpen/

[2] 2020. *Github android topic website.* https://github.com/topics/android

[3] 2020. *Github topic: Android.* https://github.com/topics/android

[4] 2020. *REST API v3.* https://developer.github.com/v3

[5] 2020. *Smartphone Market Share.* https://www.idc.com/promo/smartphone-market-share/os

[6] Waqar Ahmad, Christian Kästner, Joshua Sunshine, and Jonathan Aldrich. 2016. Inter-app communication in android: Developer challenges. In *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*. IEEE, 177–188.

[7] Kevin Allix, Tegawendé F Bissyandé, Jacques Klein, and Yves Le Traon. 2016. Androzoo: Collecting millions of android apps for the research community. In *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*. IEEE, 468–471.

[8] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, Konrad Rieck, and CERT Siemens. 2014. Drebin: Effective and explainable detection of android malware in your pocket.. In *Ndss*, Vol. 14. 23–26.

[9] Lingfeng Bao, David Lo, Xin Xia, Xinyu Wang, and Cong Tian. 2016. How Android App Developers Manage Power Consumption?-An Empirical Study by Mining Power Management Commits. In *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*. IEEE, 37–48.

[10] Haipeng Cai, Ziyi Zhang, Li Li, and Xiaoqin Fu. 2019. A Large-Scale Study of Application Incompatibilities in Android. In *The 28th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2019)*.

[11] Luis Cruz and Rui Abreu. 2019. On the Energy Footprint of Mobile Testing Frameworks. *IEEE Transactions on Software Engineering* (2019).

[12] Luis Cruz, Rui Abreu, John Grundy, Li Li, and Xin Xia. 2019. Do Energy-oriented Changes Hinder Maintainability?. In *The 35th IEEE International Conference on Software Maintenance and Evolution (ICSME 2019)*.

[13] Feng Dong, Haoyu Wang, Li Li, Yao Guo, Tegawendé F Bissyandé, Tianming Liu, Guoai Xu, and Jacques Klein. 2018. FraudDroid: Automated Ad Fraud Detection for Android Apps. In *The 26th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2018)*.

[14] Jun Gao, Li Li, Tegawendé F Bissyandé, and Jacques Klein. 2019. On the Evolution of Mobile App Complexity. In *The 24th International Conference on Engineering of Complex Computer Systems (ICECCS 2019)*.

[15] Jun Gao, Li Li, Pingfan Kong, Tegawendé F Bissyandé, and Jacques Klein. 2019. Understanding the Evolution of Android App Vulnerabilities. *IEEE Transactions on Reliability (TRel)* (2019).

[16] Sarra Habchi, Naouel Moha, and Romain Rouvoy. 2019. The rise of Android code smells: who is to blame?. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, 445–456.

[17] Geoffrey Hecht, Naouel Moha, and Romain Rouvoy. 2016. An empirical study of the performance impacts of android code smells. In *Proceedings of the international conference on mobile software engineering and systems*. 59–69.

[18] Pingfan Kong, Li Li, Jun Gao, Tegawendé F Bissyandé, and Jacques Klein. 2019. Mining Android Crash Fixes in the Absence of Issue- and Change-Tracking Systems. In *The 28th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2019)*.

[19] Pingfan Kong, Li Li, Jun Gao, Kui Liu, Tegawendé F Bissyandé, and Jacques Klein. 2018. Automated Testing of Android Apps: A Systematic Literature Review. *IEEE Transactions on Reliability* (2018).

[20] Daniel E Krutz, Mehdi Mirakhorli, Samuel A Malachowsky, Andres Ruiz, Jacob Peterson, Andrew Filipski, and Jared Smith. 2015. A dataset of open-source android applications. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*. IEEE, 522–525.

[21] Li Li. 2017. Mining androzoo: A retrospect. In *The Doctoral Symposium of 33rd International Conference on Software Maintenance and Evolution (ICSME-DS 2017)*.

[22] Li Li, Alexandre Bartel, Tegawendé F Bissyandé, Jacques Klein, Yves Le Traon, Steven Arzt, Siegfried Rasthofer, Eric Bodden, Damien Octeau, and Patrick McDaniel. 2015. Iccta: Detecting inter-component privacy leaks in android apps. In *Proceedings of the 37th International Conference on Software Engineering-Volume 1*. IEEE Press, 280–291.

[23] Li Li, Tegawendé F Bissyandé, and Jacques Klein. 2019. Rebooting Research on Detecting Repackaged Android Apps: Literature Review and Benchmark. *IEEE Transactions on Software Engineering (TSE)* (2019).

[24] Li Li, Tegawendé F Bissyandé, Mike Papadakis, Siegfried Rasthofer, Alexandre Bartel, Damien Octeau, Jacques Klein, and Le Traon. 2017. Static analysis of android apps: A systematic literature review. *Information and Software Technology* 88 (2017), 67–95.

[25] Li Li, Tegawendé F Bissyandé, Haoyu Wang, and Jacques Klein. 2018. CiD: Automating the Detection of API-related Compatibility Issues in Android Apps. In *The ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2018)*.

[26] Li Li, Jun Gao, Tegawendé F Bissyandé, Lei Ma, Xin Xia, and Jacques Klein. 2020. CDA: Characterising Deprecated Android APIs. *Empirical Software Engineering (EMSE)* (2020).

[27] Li Li, Jun Gao, Médéric Hurier, Pingfan Kong, Tegawendé F Bissyandé, Alexandre Bartel, Jacques Klein, and Yves Le Traon. 2017. AndroZoo++: Collecting Millions of Android Apps and Their Metadata for the Research Community. *arXiv preprint arXiv:1709.05281* (2017).

[28] Li Li, Daoyuan Li, Tegawendé F Bissyandé, Jacques Klein, Yves Le Traon, David Lo, and Lorenzo Cavallaro. 2017. Understanding Android App Piggybacking: A Systematic Study of Malicious Code Grafting. *IEEE Transactions on Information Forensics & Security (TIFS)* (2017).

[29] Tianming Liu, Haoyu Wang, Li Li, Xiapu Luo, Feng Dong, Yao Guo, Liu Wang, Tegawendé F Bissyandé, and Jacques Klein. 2020. MadDroid: Characterising and Detecting Devious Ad Content for Android Apps. In *The Web Conference 2020 (WWW 2020)*.

[30] William Martin, Federica Sarro, Yue Jia, Yuanyuan Zhang, and Mark Harman. 2016. A survey of app store analysis for software engineering. *IEEE transactions on software engineering* 43, 9 (2016), 817–847.

[31] Guozhu Meng, Yinxing Xue, Jing Kai Siow, Ting Su, Annamalai Narayanan, and Yang Liu. 2017. Androvault: Constructing knowledge graph from millions of android apps for automated analysis. *arXiv preprint arXiv:1711.07451* (2017).

[32] Maleknaz Nayebi, Konstantin Kuznetsov, Paul Chen, Andreas Zeller, and Guenther Ruhe. 2018. Anatomy of functionality deletion: an exploratory study on mobile apps. In *Proceedings of the 15th International Conference on Mining Software Repositories*. 243–253.

[33] Fabio Palomba, Dario Di Nucci, Annibale Panichella, Andy Zaidman, and Andrea De Lucia. 2017. Lightweight detection of android-specific code smells: The adoctor project. In *2017 IEEE 24th international conference on software analysis, evolution and reengineering (SANER)*. IEEE, 487–491.

[34] Simone Scalabrino, Gabriele Bavota, Mario Linares-Vásquez, Michele Lanza, and Rocco Oliveto. 2019. Data-driven solutions to detect API compatibility issues in Android: an empirical study. In *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*. IEEE, 288–298.

[35] Lili Wei, Yepang Liu, and Shing-Chi Cheung. 2019. Pivot: learning API-device correlations to facilitate Android compatibility issue detection. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 878–888.

[36] Yanjie Zhao, Haoyu Wang, Lei Ma, Yuxin Liu, Li Li, and John Grundy. 2019. Knowledge Graphing Git Repositories: A Preliminary Study. In *IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER 2019)*.

[37] Yajin Zhou and Xuxian Jiang. 2012. Dissecting android malware: Characterization and evolution. In *2012 IEEE symposium on security and privacy*. IEEE, 95–109.