2015 12th Working Conference on Mining Software Repositories

A Dataset of Open-Source Android Applications

Daniel E. Krutz, Mehdi Mirakhorli, Samuel A. Malachowsky, Andres Ruiz,
Jacob Peterson, Andrew Filipski, and Jared Smith
Rochester Institute of Technology, Rochester, NY, USA
{dxkvse, mxmvse, samvse, ajr2546, jrp9988, abf1932, jps6773}@rit.edu

***Abstract*—Android has grown to be the world's most popular mobile platform with *apps* that are capable of doing everything from checking sports scores to purchasing stocks. In order to assist researchers and developers in better understanding the development process as well as the current state of the apps themselves, we present a large dataset of analyzed open-source Android applications and provide a brief analysis of the data, demonstrating potential usefulness. This dataset contains 1,179 applications, including 4,416 different versions of these apps and 435,680 total commits. Furthermore, for each app we include the analytical results obtained from several static analysis tools including Androguard, Sonar, and Stowaway.**

In order to better support the community in conducting research on the security characteristics of the apps, our large analytical dataset comes with the detailed information including various versions of AndroidManifest.xml files and synthesized information such as permissions, intents, and minimum SDK. We collected 13,036 commits of the manifest files and recorded over 69,707 total permissions used. The results and a brief set of analytics are presented on our website: http://androsec.rit.edu.

***Index Terms*—Open-source dataset, Android development, Software Engineering**

I. Introduction

Android has become an extremely popular mobile platform, and Android apps are not immune to the problems which have hindered traditional software — especially security vulnerabilities, high maintenance costs, and bugs. Understanding how software is created and maintained is paramount in determining how to produce it faster, cheaper, and of higher quality. One way to gain valuable insight into the development process is to examine existing projects: source code, how the app has evolved over time, and attributes such as its security, defects, and size. App source code may be analyzed using static analysis tools, providing data about the software's security risk level, possible defects, or even lack of adherence to coding standards. Studying version control history can provide information about how the app was created — especially through the examination of commit messages, who made the commit, and when it was made.

We have created a dataset of 1,179 open-source Android apps with 435,680 version control commits through the mining of F-Droid[1], a repository of Android apps of various sizes and categories. In this paper we discuss (i) the data collection and analysis process, (ii) web portal developed to share the data as well as analytical results in an easy to use manner, and (iii) different characteristics and attributes of this dataset. Our goal is to create a publicly available dataset which researchers can utilize to conduct more comprehensive experiments on Android app development.

[1]https://f-droid.org

II. Related Work

While we are unaware of any projects which have gathered such a substantial amount of Android data, performed various types of static analysis upon it, and made is as publicly available as we did, there are several existing commercial websites which do provide metics about Android applications. Appannie[2], AppBrain[3], and AppZoom[4] contain analytical and statistical information about hundreds of thousands of Android apps in a robust and easy to use online format. They do not appear to make their data fully transparent or examine the version histories as we have done.

A large number of previous works have analyzed version control systems for various software engineering purposes. Eyolfson et al. [5] examined the effects that developer experience, and date and time of commit had on the bugginess of an application. Buse and Weimer [2] used commit messages to automatically document program changes. While we do not perform any data analysis in this paper, the previous use of version control information for software engineering research demonstrates the importance and relevance of our dataset.

III. Dataset Construction

Our dataset was built in two primary phases: the data collection process which included gathering source code and version control information from the F-Droid repository, and an analysis using several static analysis tools. An overview of the process is shown in Figure 1.

A. Collection Process

For the first phase we built a scraper tool to collect data from the F-Droid repository, extracting information from each app such as meta-data (name, description, and version), the source code of each major version, and its most recent apk (Android application) file. Additionally, we collected version control information such as the committer user name, commit time, and commit messages, ensuring all data was tagged with the version number to allow analysis over time.

[2]http://www.appannie.com
[3]http://www.appbrain.com
[4]http://www.appszoom.com

978-0-7695-5594-2/15 $31.00 © 2015 IEEE
DOI 10.1109/MSR.2015.79

TABLE I: Data Overview

	Value	Count
Totals	**Apps**	1,179
	Versions	4,416
	Commiters	4,535
	Committs	435,680
Largest	**Versions**	48
	Committers	202
	Commits	65,110

TABLE II: Version Counts

Min Versions	Count
3	466
5	258
10	97
15	54
20	26
25	16
30+	12

TABLE III: Commit Keywords

Keyword	Version Count
Fix	81,558
Bug	21,380
Version	14,031
Hack	1,707
Performance	809
Permission	700
Failure	512
Security	148

TABLE IV: Committer Count

		Min Commit Count		
		10	**25**	**50**
Count of	**Apps**	118	53	23
	Categories	13	13	11
Avg	**Commiter Count**	35	60	94
	Commit Count	2,841	5,233	9,240

TABLE V: Permission Counts

Permission	Count
INTERNET	9,049
WRITE_EXTERNAL_STORAGE	6,518
ACCESS_NETWORK_STATE	5,778
WAKE_LOCK	3,886
RECEIVE_BOOT_COMPLETED	3,402

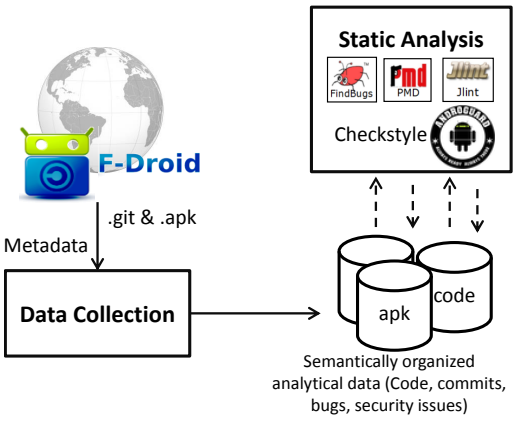

Fig. 1: Collection & Analysis Process

Once major version history had been established, each appropriate apk was downloaded from the version control repository and analyzed for additional metadata. The AndroidManifest.xml file (also available in the dataset) includes information on permissions, intents, target sdk, and minimum sdk, which were extracted using a second script.

B. Static Analysis

Once collection was complete, we continued with analysis, running a variety of static analysis tools on the app's source code. Androrisk[5] and Stowaway [6] were used to analyze the apk files, and Sonar was used on the extracted source code.

Stowaway: Android developers operate under a permission-based system where apps must be granted access (by the user and operating system) to various areas of functionality before they may be used. Examples include GPS location information, contacts, or the ability to make a phone call. If an app attempts to perform an operation to which it does not have permission, a *SecurityException* is thrown. Stowaway discovers these permission-gaps — the over-permission and under-permission rate of an application.

We selected Stowaway because it is able to state explicitly what over-permissions and under-permissions are present using a static-analysis based approach (not requiring an Android device or emulator). Stowaway has also demonstrated its effectiveness in existing research [6]. Permlyzer [8], a more modern permission detection tool, was not used because its authors have not made it available for download.

Androrisk: A component of the Androguard reverse engineering tool, Androrisk calculates a risk indicator of an app based upon various settings and permissions requested by the application. The presence of permissions that may send an SMS, place a call, or access the internet, for example, have varying weights which will elevate the application's risk level. The total reported security risk score for each application is recorded and available.

We chose Androrisk because it is freely available and open-source (allow others to confirm our findings), it has the ability to quickly process a large number of apps (via static analysis), and the AndroGuard library (of which Androrisk is a component) has already been used in existing research [4].

Sonar[6]: Sonar is a source code analysis tool which covers the *7 axes* of code quality: architecture and design, comments, coding rules, potential bugs, complexity, unit tests, and duplications. Sonar was chosen for the wide range of code metrics and defect analysis that it provides; in addition having to its own analysis components, it integrates three of the most popular static source code analysis tools: FindBugs[7], Checkstyle[8], and PMD[9]. FindBugs uses static analysis to identify hundreds of different error types within Java source code, allowing us to find correlations between bugs and other recorded metrics within applications. Checkstyle determines how well Java source code adheres to coding rules and

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

[6]http://www.sonarqube.org
[7]http://findbugs.sourceforge.net
[8]http://checkstyle.sourceforge.net
[9]http://pmd.sourceforge.net

standards, and PMD analyzes code to identify bad practices that may cause a more inefficient and harder to maintain codebase.

IV. Analytics & Data Sharing

The extracted data and analytical results are shared through our website, http://androsec.rit.edu. Our goal is to provide several ways for users or researchers to efficiently access and use this data. A general user can simply use this web portal to retrieve analytical information about a specific app. As an example, they may wish to see if a specific application version has more over-permissions or potential bugs as compared to a previous version. A researcher can utilize more advanced features of the site, such as downloading the entire dataset as well as analytical results.

The entire SQLite database, which contains the static analysis results and other data from the version control systems, is also available for download on our project website along with all collected .apk files.

A. Exploring the Dataset

In order to provide an understanding of the dataset, we have created some metrics exploring the depth and breadth of the applications and metadata. An overview of the total number of unique apps, versions, committers, and commits along with the highest number of versions, committers and commits for any collected app is shown in Table I.

Table II displays the number of apps which have at least a specific number of versions, which is shown in the *Min Versions* column. Our dataset contains 466 apps which have at least 3 developer defined versions, and 12 apps which have 30 or more versions.

Table IV displays the number of apps and categories which have an average number of 10, 25, and 50 unique committers. A unique committer is defined as a commit made by someone using a unique name in the version control system for a specific app. Categories are the types of groups the app has been defined as belong to — examples include office, internet, navigation, and games.

Version control commit messages have been used in a wide range of software engineering research [1] including assisting in the automatic documentation of program changes [2] and helping with the identification of bug fixing patches [3], which typically searches for keywords such as “bug” and “fix” in order to find bug fixing commits [7]. For each of the 435,680 recorded commits, we have stored the author’s commit message. Examples of these messages include “Added progress bar to notification”, “Fix a bug with the whitelist”, and “Fix key input in credits sequences.” Table III displays some keywords and the number of commits each keyword appeared in.

Information about permissions is one of the most important data-points and may be used to determine how permission settings evolve over time, which permissions are most prominently used, and if outdated permissions are still used instead of newer ones. Table V displays the top 5 most commonly used permissions in all AndroidManifest.xml commits along with the number of times they appeared. Complete results are available in our database for further analysis.

B. Analytical Results

The project website contains several reports and information pages, and users may choose to download the entire dataset. We display a wide range of aggregate information by category for all apps including the top over-permissions, most vulnerable apps according to Androrisk, and coding standards violations. As an example, Figure 2 illustrates the number of over-permission issues by category.

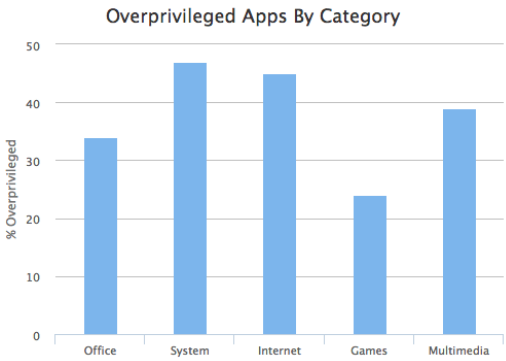

Fig. 2: Example Aggregate Data

Furthermore, a user can use the web portal and search for information about a specific app, snapshot of this search is shown in Figure 3. We provide data about individual app versions as collected from static analysis including over and under permissions, Androrisk vulnerability scores, defect analysis, and code complexity. Figure 4 shows defect information about a sample app (FBReader) over the course of multiple versions.

Search: FB

Name	# Versions	Current Version	Repo Type
FBReader TTS+ Plugin	12	3.5.4	git
FBReader TTS plugin	2	1.2	git
FBReader	37	1.8.2	git
FBReader Calibre connector	2	1.2	git

Showing 1 to 4 of 4 entries (filtered from 1,179 total entries)

Fig. 3: App Search

As shown in Figure 5, users may also explore the data by writing their own queries against the dataset right on the webpage.

C. Enabled Research

A dataset such as ours has a vast array of possible uses such as helping to better understand the development process of individual Android applications or studying dominant paradigms

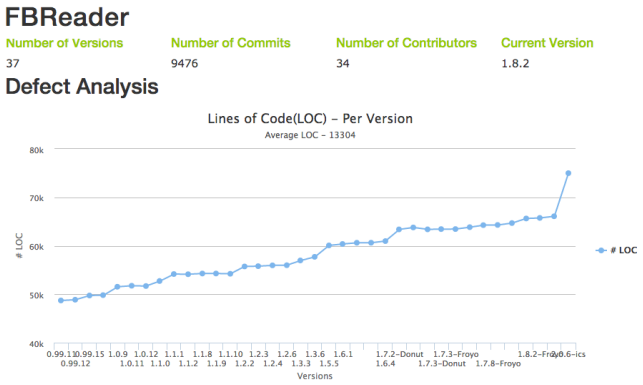

Fig. 4: Information About Specific App

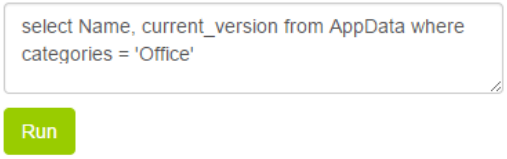

Fig. 5: Webpage Search Query

in app development at a more aggregate level. We next provide exemplar usage scenarios for such data.

Facilitate research on mining software repositories Researchers can easily download all of the raw data as well as analytical results and conduct research experiments. An example of empirical software engineering research that could be assisted by our dataset is the exploration of the evolution of Android applications. Overall, we collected 4,416 versions of 1,179 apps, however the number of collected versions for each app widely varied. For example, 514 apps only had 1 defined version while one of the apps had 48 total versions. Future researchers may be able to use the included commit message information to study different characteristics of apps.

Our version control history not only includes the log message, but also the committer and the time of commit, which has been previously used in wide range of mining software repositories research [1]–[3], [5].

Information collected about the AndroidManifest.xml commits can be used to determine how the app's settings and permission levels change and evolve over the time.

Benchmark Dataset The collected data and the static analysis results can be used as benchmark datasets, allowing other researchers to compare their homemade static analysis results with our collected and analyzed data. This is especially useful since we are not attempting to present any specific findings or tools in our work, only data, eliminating any reason to exclude or bias information contained within the dataset.

V. Limitations & Future Work

While we feel that our dataset is robust and quite useful for a variety of areas of future research, it does have some limitations and areas that will be improved upon. In the future, new static analysis tools may be added to examine the apps. Since we are storing the version control histories and source code locally, running these tools retroactively against previous versions of apps should require little effort.

While we feel that collecting more than 1,000 apps with more than 4,000 versions and over 430,000 total commits represents a substantially sized dataset, there are over 1.4 million[10] apps available on GooglePlay, so our collection represents only a very minor portion of all apps. Additionally, we only analyzed free apps, excluding paid apps in our dataset.

Future improvements will be made to the website not only making it more usable, but also displaying the data in a more user friendly manner as well as allowing more customizable reports.

VI. Conclusion

A dataset of Android applications with the results of the static analysis tools we have created is an important tool for understanding how Android applications are developed and maintained. The metrics we have provided are useful for understanding potential correlations between the various collected data metrics, not only for individual apps, but for apps as an aggregate as well. The collected data is publicly available on the project website: http://androsec.rit.edu

References

[1] A. Bachmann, C. Bird, F. Rahman, P. Devanbu, and A. Bernstein. The missing links: bugs and bug-fix commits. In *Proceedings of the eighteenth ACM SIGSOFT international symposium on Foundations of software engineering*, pages 97–106. ACM, 2010.

[2] R. P. Buse and W. R. Weimer. Automatically documenting program changes. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, ASE '10, pages 33–42, New York, NY, USA, 2010. ACM.

[3] V. Dallmeier and T. Zimmermann. Extraction of bug localization benchmarks from history. In *Proceedings of the Twenty-second IEEE/ACM International Conference on Automated Software Engineering*, ASE '07, pages 433–436, New York, NY, USA, 2007. ACM.

[4] 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.

[5] J. Eyolfson, L. Tan, and P. Lam. Do time of day and developer experience affect commit bugginess? In *Proceedings of the 8th Working Conference on Mining Software Repositories*, MSR '11, pages 153–162, New York, NY, USA, 2011. ACM.

[6] 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.

[7] Y. Tian, J. Lawall, and D. Lo. Identifying linux bug fixing patches. In *Proceedings of the 34th International Conference on Software Engineering*, ICSE '12, pages 386–396, Piscataway, NJ, USA, 2012. IEEE Press.

[8] 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.

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