2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)

An Empirical History of Permission Requests and Mistakes in Open Source Android Apps

Gian Luca Scoccia*, Anthony Peruma†, Virginia Pujols†, Ben Christians†, Daniel E. Krutz†
*Gran Sasso Science Institute, L'Aquila, Italy
gianluca.scoccia@gssi.it
†Rochester Institute of Technology, Rochester, NY, USA
{axp6201, vp2532, bbc7909, dxkvse}@rit.edu

***Abstract*—Android applications (apps) rely upon proper permission usage to ensure that the user's privacy and security are adequately protected. Unfortunately, developers frequently misuse app permissions in a variety of ways ranging from using too many permissions to not correctly adhering to Android's defined permission guidelines. The implications of these *permission-issues* (possible permission problems) can range from harming the user's perception of the app to significantly impacting their privacy and security. An imperative component to creating more secure apps that better protect a user's privacy is an improved understanding of how and when these issues are being introduced and repaired.**

While there are existing permissions-analysis tools and Android datasets, there are no available datasets that contain a large-scale empirical history of permission changes and mistakes. This limitation inhibits both developers and researchers from empirically studying and constructing a holistic understanding of permission-related issues. To address this shortfall with existing resources, we created a dataset of permission-based changes and permission-issues in open source Android apps. Our unique dataset contains information from 2,002 apps with commits from 10,601 unique committers, totaling 789,577 commits. We accomplished this by mining app repositories from F-Droid, extracting their version and commit histories, and analyzing this information using two permission analysis tools. Our work creates the foundation for future research in permission decisions and mistakes. Complete project details and data is available on our project website: https://mobilepermissions.github.io

***Index Terms*—Mobile Permissions, Mobile Software Engineering, Software Repository Mining**

I. Introduction

The permissions used by apps are an imperative component of ensuring the privacy and security of the user. Due to their critical importance, the permissions utilized by an app should be a first-class concern for the developer [6], [13], [23], [24]. Unfortunately, research has shown that developers frequently misuse permissions for a variety of reasons, thus creating unnecessary privacy and security risks for the user [15], [11], [18], [16], [14]. Examples of *permission-issues* (possible permission problems) include the granting of too many permissions (*over-permissions*) to an app. Over-permissions are considered security risks since they do not adhere to the *principle of least privilege* and unnecessarily increase an app's attack surface [15]. Another example of permission-misuse includes not adhering to Android's defined permission best practices [6]. This can have a wide range of negative implications including adversely impacting the functionality of the app and hurting the user's general usage experience. To address these challenges, it is imperative for developers and researchers to better understand how and when permissions are modified in apps, along with characteristics about the developers both creating and repairing the permission-issue from a qualitative and quantitative perspective. This knowledge will enable developers and researchers to correctly understand and address these issues from a qualitative and quantitative perspective. We define *permission events* as the addition or removal of permission requests or permission problems.

Unfortunately, developers and researchers are inhibited from empirically understanding permission usage and permission-issues by the lack of an existing high-quality dataset containing this necessary information. To address this limitation, we created a permission usage and permission-issue evolutionary history of 2,002 open source Android apps. This dataset will enable the better understanding of the permission evolution process. In addition to extracting a diverse range of commit information and the history of requested permissions, collected apps were analyzed using two existing permission analysis tools: M-Perm [12] and P-Lint [13]. This analysis provides an empirical overview of permission-issues in the app development process, including how and when the permissions are added, who is making these permission-based decisions, and when are these mistakes added and removed. We provide a publicly available dataset that researchers may utilize to conduct more comprehensive experiments on Android permissions and their evolution during the app development process.

The rest of the paper is organized as follows: Section II describes the construction of the dataset. Section III provides details on the collected data and Section IV describes some enabled research from our dataset. Section V discusses related works and Section VI describes future improvements and concludes our work.

II. Dataset Construction

Several steps were used in the construction of our dataset. An overview of our data construction process is shown in Figure 1 and each of the utilized steps are described below.

1) Create catalog of F-Droid app projects F-Droid [4] is a catalog of open source Android apps, and contains links to thousands of Android app GitHub repositories. These projects

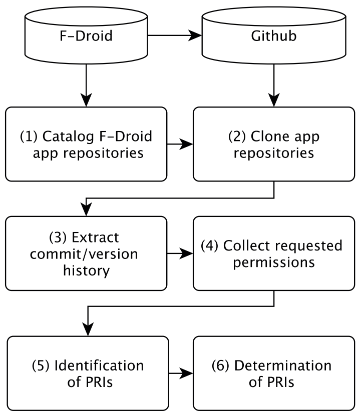
F-Droid
Github
(1) Catalog F-Droid app repositories
(2) Clone app repositories
(3) Extract commit/version history
(4) Collect requested permissions
(5) Identification of PRIs
(6) Determination of PRIs

Fig. 1. Our Data Collection and Analysis Workflow

range from small apps that are infrequently updated to large and popular apps. We selected F-Droid as our source of apps due to the diversity of apps in its catalog and for its use in prior research works [21], [20], [9]. Overall, our dataset contains information regarding 2,002 apps.

2) Clone app repositories Our next step is to clone each of the collected GitHub repositories from the F-Droid catalog and record app metadata. Some of this metadata information includes the name, description, category and repository URL of each app. Overall, we extracted 789,577 total commits.

3) Extract commit/version history of each app After cloning each repository, we extracted: (I) Each commited app version; (II) Commit details of each app, including commiter, timestamp, and commit message; (III) All commited `Androidmanifest.xml` file versions. Section III provides more information regarding collected data.

4) Collect Requested Permissions We recorded the requested permissions for each app version by collecting them from the extracted `Androidmanifest.xml` files. Using this information, we were able to determine the commits that added and removed each permission request, therefore enabling us to correlate this with developers who were making permission-based decisions. We identify *dangerous* [1], *normal* [2] and *custom* [3] permissions in our collected data.

5) P-Lint & M-Perm analysis We next examined each of the extracted app versions for permission-related issues (PRIs) using two open source analysis tools: M-Perm [12] and P-Lint [13]. Only app versions using the current run-time permission model (Android 6.0+) were analyzed, as these tools are only capable of analyzing the current run-time permission model. Even though other tools (Stowaway [15] and PScout [10]) could have been adopted to analyze remaining apps, we believe that the permission-issues of a long deprecated permission model would not be of interest to current researchers. We will next describe the analysis tools used in our study.

M-Perm M-Perm discovers cases of over and under permissions by combining static and dynamic analysis to identify cases where too many (over-permissions) or too few (under-permissions) permissions were requested by the app. Over-permissions have been widely examined in existing works and have been found to be detrimental to the security and privacy of the user [15], [25], [27].

P-Lint: This tool detects *permission smells* [13], which are instances of developers failing to adhere to Android's defined permissions best practices [6]. Permission smells are similar to code smells in that they are indications of possible permission-issues, but may have a benign impact on the application. An example of a permission smell is the improper implementation of the *checkSelfPermission()* function according to Android's recommended guidelines [7]. Although this type of issue will not invariably lead to a problem in every occurrence, it is potentially detrimental to the user experience. P-Lint identifies 16 forms of permission-smells.

To evaluate M-Perm and P-Lint, we built a small oracle of five simple Android apps. We next created a second version of these apps with each containing at least one permission-issue that would be detected by both M-Perm and P-Lint. We then ran M-Perm and P-Lint against these apps and found that they obtained a precision and recall value of 1.0. While simplistic, these results provided confidence in the ability of these tools to accurately discover permission-issues. These oracle apps are available on our project website [5]. After this evaluation phase provided confidence in the ability of the tools, we next used M-Perm and P-Lint to analyze each of the extracted commits for each of the collected apps, and recorded instances of permissions-misuse.

6) Determination of permission-issues After the detection of permission-issues, we determine the commits that *created* and *fixed* each of them. This is a non-trivial task as identifying these issues involves much more analysis than merely examining each committed version. For example, to determine the commit that repaired a permission-issue, we must find instances where a subsequent commit does not contain the same permission-issue that was found in the previous commit. The following statuses were used to define each discovered permission-issue, and provide further information about how they were identified:

- **New.** The permission-issue did not exist in the previous version of the file, and is therefore new. This was found by examining the previous commits to determine the commit that introduced this permission-issue.
- **Exist.** The permission-issue existed in previous and subsequent committed versions.
- **Fix.** The permission-issue was found in an examined commit, but did not appear in the subsequent commit. This subsequent commit that the issue did not appear in was determined to be the 'fix' commit.

These recorded events are complemented by an array of further collected information about the permission event, some of which includes: the commiter, the date/time of the com-

TABLE I
OVERVIEW OF COLLECTED DATA

Total Count	**Value**
Cloned app repositories	2,002
Commits	789,577
Unique committers	10,601
Recorded permission instances	3,614,165
Permission changes	63,006
Unique *Custom* [3] Permissions	295

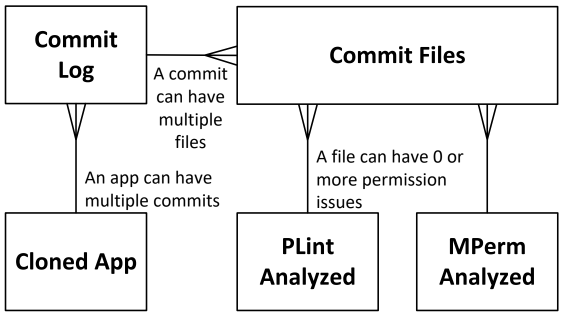
Commit Log
A commit can have multiple files
Commit Files
An app can have multiple commits
A file can have 0 or more permission issues
Cloned App
PLint Analyzed
MPerm Analyzed

Fig. 2. An overview of the database schema

mit, the commit message, and even other files that were changed with the commit. This supporting information will assist the researcher in creating a more holistic perspective of the circumstances related to the permission event. It took approximately one month to extract about 2 TB of source files that were associated with project commits. A further 2.5 weeks were required to run MPerm and PLint on this dataset.

III. Analytics and Data Sharing

The objective of this work is to assist developers and researchers in better understanding permission events during the development process of an app. All data is available on the project website [5] in the form of an SQLite database, along with other relevant project information.

A. Data Overview

To create a fundamental understanding of the dataset, we will next provide an overview of some of the generated and collected data. Table I shows a high-level overview of some of the collected and generated data. A simplified view of the schema is presented in Figure 2; more details are available on the project website.

B. Individual App Example: VLC

We will next demonstrate the ability of our dataset to provide permission-related information about a single application. 'VLC' [8] is a popular app used to play media files, and is one of the apps included in our dataset. We selected VLC as our example app since it has a large number of commits

TABLE II
SAMPLE ANALYSIS INFORMATION FOR VLC APP

Item	**Value**
Most common over-permission	READ_PHONE_STATE
Most requested 'Dangerous'[1] permission	READ_PHONE_STATE
Most requested 'Normal'[2] permission	INTERNET
'Normal' permissions count (all commits)	54,190
'Dangerous' permissions count (all commits)	13,168
Total number of developers	124
Unique Custom permissions	3
Commits with permission changes	368

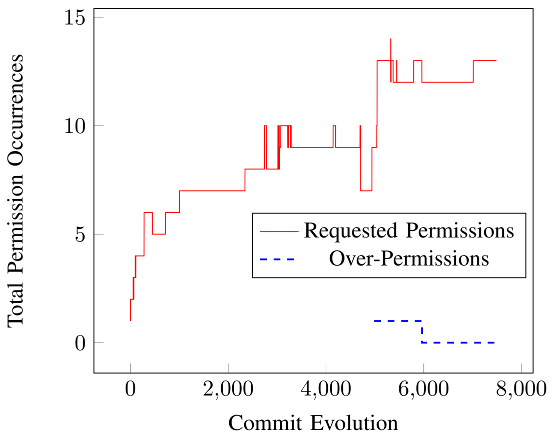

Fig. 3. Evolution of requested permissions and over-permissions in VLC app

and is popular (100M+ downloads on the Google Play Store). Table II displays permission information in the VLC app.

Developers and researchers may wish to examine the evolution of permission events during the lifecycle of the app. Figure 3 displays the evolution of permission requests and over-permissions during the development history of the VLC app. In this example, although there were deviations, the number of requested permissions generally grew as the app evolved. When the app switched to the Android run-time permission model, there were initially several observed over-permissions. However, these were removed at approximately the #6,000 commit mark. Due to the popularity of VLC and the size and anticipated quality of the development team, it is expected that an app with these developer characteristics would exhibit a low number of over-permissions. The number of over-permissions is not shown until the app adopted the run-time permission model.

Using the VLC app, we will next demonstrate the ability of our dataset to track the history of a permission-issue during the development of an app. Table III displays information about an identified over-permission during the app's development. Discovered information includes the creator of the issue, when it was introduced, how long it existed for (in terms of time duration and commits), along with who repaired the issue and when it was repaired. This represents just one of the many possible empirical examinations that are enabled through the use of our dataset. To not single out the developer who made the mistake, we refer to the developer as "Developer X". Their actual information may be found in our dataset.

TABLE III
EXAMPLE OF PERMISSION-ISSUE IN VLC APP

Item	**Value**
Over-Permission	READ_PHONE_STATE
Introducing Developer	Developer 'X'
Date of Introduction	4/19/16
Duration (Days)	128
Duration (Commits)	636
Fixing Developer	Developer 'X'
Date of Repair	8/25/16

IV. Enabled Research

Our dataset will support a wide-range of research regarding permission events. Some of the enabled research includes:

- Investigate the characteristics of developers who make important permission-based decisions throughout the development life-cycle of the app.
- Understand the correlation between permission smells [13] and code smells occurring in the app, along with the type of developers contributing to the introduction and removal of the smells.
- Further understand the characteristics of permission decisions and permission-issues.
- Study the correlation among permission events and other occurrences during the development life cycle.
- Expand our knowledge about usage of permissions during app development.
- Research the effect of the emotional tone of developers when conducting permission-related tasks, through sentiment analysis on commit messages and collected complementary information.

V. Related Work

One of the earliest studies on permissions misuse was conducted by Felt *et al.* [15], and found that 36% of apps suffered from at least one over-permission. Tang [25] examined 10,710 apps from Google Play and found that 76% of the apps contain at least one over-privilege. Calciati *et al.* [11] studied how apps evolve over time and discovered that apps typically request more permissions over time and that the removal of permissions does not typically imply the loss of functionality. Our work differs from this in that we additionally focus on developer tendencies and who was actually making the permissions-based decisions and mistakes in the app development and maintenance process.

Allix *et al.* [9] crawled multiple sources to build a dataset of over 3,000,000 unique Android APK files. However, this dataset does not contain repository information or the evolutionary history of the app. Grano *et al.* [19] created a dataset of user reviews from Google Play for 395 open source apps. Munaiah *et al.* [22] reversed engineered over 64,000 apps from Google Play and other sources to perform an analysis of security-related decisions made by app developers.

Several vendors such as App Annie[1], AppBrain[2] and AppFollow[3] provide commercial datasets of metrics related to Android apps. However, most of the data in their datasets are related to app store analytics and not source code. Krutz *et al.* [21] also mined F-Droid to gather information about existing apps. However, this work significantly differs from ours in several key areas: (I) They did not analyze the history of the app from the developer's perspective, and thus did not provide a more in-depth and all-encompassing view into the evolution of the app; (II) They examined a smaller number of apps; 1,179 vs. 2,002 ; (III) They only analyzed the apps using several high-level security and quality tools, whereas our work examines additional aspects such as permission-issues.

Taylor *et al.* [26] used snapshots of requested permissions from the Google Play store to examine the evolution of app permission usage with each app release. This work found that apps were progressively requesting more permissions over time. This work differs from ours in that we examine apps at a much more granular level, and do not only examine when permission requests are being made. We also examine who is making the permission requests, along with examining permission-issues from a variety of perspectives. Geiger *et al.* [17] created 'AndroidTimeMachine', using data from both GitHub and Google Play. This dataset differs from ours in that they do not specifically target permission information or permission related issues in their work.

VI. Conclusions and Future Work

While this represents the first known public dataset of empirically generated permission changes and issues, there are several improvements that may be made. The dataset only contains projects that are Git based. Future work will integrate apps hosted on other systems such as Subversion to enable the collection and analysis of additional apps. The dataset is currently available as a standalone SQLite database. However, our goal is to have the data available in a SaaS model. This will enable researchers and developers to access our data directly through APIs and perform queries using our public web application.

A key component of developing better software is to understand how it is being created. Our work provides a foundational dataset that developers and researchers may use to better understand the app creation and evolution process, specifically in regards to permissions. Several of the benefits provided by this dataset include: (I) Increased understanding of developer tendencies and mistakes in regards to permissions (II) Knowledge of utilized permissions and mistakes in Android apps (III) Better understanding of permission evolution in apps. All data is publicly available on the project website: **https://mobilepermissions.github.io**

[1]https://www.appannie.com/
[2]https://www.appbrain.com/
[3]https://www.appfollow.io/

References

[1] Android permissions overview: Dangerous permissions. https://developer.android.com/guide/topics/permissions/overview#dangerous-permission-prompt.

[2] Android permissions overview: Normal permissions. https://developer.android.com/guide/topics/permissions/overview#normal_permissions.

[3] Define a custom app permission. https://developer.android.com/guide/topics/permissions/defining.

[4] F-droid. https://f-droid.org/en/.

[5] Investigating permission issues in open-source android apps. https://mobilepermissions.github.io/.

[6] Permissions best practices. https://developer.android.com/training/permissions/best-practices.html.

[7] Request app permissions. https://developer.android.com/training/permissions/requesting.

[8] Vlc-android. https://code.videolan.org/videolan/vlc-android.

[9] K. Allix, T. F. Bissyand, J. Klein, and Y. L. Traon. Androzoo: Collecting millions of android apps for the research community. In *2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR)*, pages 468–471, May 2016.

[10] K. W. Y. Au, Y. F. Zhou, Z. Huang, and D. Lie. Pscout: Analyzing the android permission specification. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, CCS '12, 2012.

[11] P. Calciati and A. Gorla. How do apps evolve in their permission requests?: A preliminary study. In *Proceedings of the 14th International Conference on Mining Software Repositories*, MSR '17, pages 37–41. IEEE Press, 2017.

[12] P. Chester, C. Jones, M. W. Mkaouer, and D. E. Krutz. M-perm: A lightweight detector for android permission gaps. In *2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, pages 217–218, 2017.

[13] C. Dennis, D. E. Krutz, and M. W. Mkaouer. P-lint: A permission smell detector for android applications. In *2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, pages 219–220, May 2017.

[14] P. Faruki, A. Bharmal, V. Laxmi, V. Ganmoor, M. S. Gaur, M. Conti, and M. Rajarajan. Android security: a survey of issues, malware penetration, and defenses. *IEEE communications surveys & tutorials*, 17(2):998–1022, 2015.

[15] 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.

[16] X. Gao, D. Liu, H. Wang, and K. Sun. Pmdroid: Permission supervision for android advertising. In *2015 IEEE 34th Symposium on Reliable Distributed Systems (SRDS)*, pages 120–129, Sept 2015.

[17] F.-X. Geiger, I. Malavolta, L. Pascarella, F. Palomba, D. Di Nucci, and A. Bacchelli. A graph-based dataset of commit history of real-world android apps. In *Proceedings of the 15th International Conference on Mining Software Repositories*, MSR '18, pages 30–33, New York, NY, USA, 2018. ACM.

[18] M. C. Grace, W. Zhou, X. Jiang, and A.-R. Sadeghi. Unsafe exposure analysis of mobile in-app advertisements. In *Proceedings of the Fifth ACM Conference on Security and Privacy in Wireless and Mobile Networks*, WISEC '12, pages 101–112, New York, NY, USA, 2012. ACM.

[19] G. Grano, A. Di Sorbo, F. Mercaldo, C. A. Visaggio, G. Canfora, and S. Panichella. Android apps and user feedback: A dataset for software evolution and quality improvement. In *Proceedings of the 2Nd ACM SIGSOFT International Workshop on App Market Analytics*, WAMA 2017, pages 8–11, New York, NY, USA, 2017. ACM.

[20] S. Habchi, G. Hecht, R. Rouvoy, and N. Moha. Code smells in ios apps: How do they compare to android? In *2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, pages 110–121, May 2017.

[21] D. E. Krutz, M. Mirakhorli, S. A. Malachowsky, A. Ruiz, J. Peterson, A. Filipski, and J. Smith. A dataset of open-source android applications. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, pages 522–525, May 2015.

[22] N. Munaiah, C. Klimkowsky, S. McRae, A. Blaine, S. A. Malachowsky, C. Perez, and D. E. Krutz. Darwin: A static analysis dataset of malicious and benign android apps. In *Proceedings of the International Workshop on App Market Analytics*, WAMA 2016, pages 26–29, New York, NY, USA, 2016. ACM.

[23] A. Peruma, J. Palmerino, and D. E. Krutz. Investigating user perception and comprehension of android permission models. In *Proceedings of the 5th International Conference on Mobile Software Engineering and Systems*, MOBILESoft '18, pages 56–66, New York, NY, USA, 2018. ACM.

[24] G. L. Scoccia, S. Ruberto, I. Malavolta, M. Autili, and P. Inverardi. An investigation into android run-time permissions from the end users' perspective. In *Proceedings of the 5th International Conference on Mobile Software Engineering and Systems, MOBILESoft@ICSE 2018, Gothenburg, Sweden, May 27 - 28, 2018*, pages 45–55.

[25] J. Tang, R. Li, H. Han, H. Zhang, and X. Gu. Detecting permission over-claim of android applications with static and semantic analysis approach. In *2017 IEEE Trustcom/BigDataSE/ICESS*, pages 706–713, Aug 2017.

[26] V. F. Taylor and I. Martinovic. To update or not to update: Insights from a two-year study of android app evolution. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, ASIA CCS '17, pages 45–57, New York, NY, USA, 2017. ACM.

[27] P. Wijesekera, A. Baokar, A. Hosseini, S. Egelman, D. Wagner, and K. Beznosov. Android permissions remystified: A field study on contextual integrity. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 499–514, 2015.