Discovering Android Data: A Large Scale Analysis of Open Source Android Apps

Daniel E. Krutz, Mehdi Mirakhorli, and Samuel A. Malachowsky

Rochester Institute of Technology, Rochester, NY, USA
{dxkvse, mxmvse, samvse}@rit.edu

Mobile applications (*apps*) are ubiquitous in today's world, allowing us to do everything from update our Facebook profile to trading of stocks at the push of a button. Unfortunately, with this great power also comes extreme dangers. Recent studies have shown that 68-77% of Android apps are susceptible to just a single SSL vulnerability. This article presents an easy to use, publicly available website and data set located to assist developers, researchers, users and educators/students with better understanding Android app security. The underlying infrastructure contains a number of security and static analysis tools which are automatically run over one thousands open source Android apps from the F-Droid repository. The Androsec portal represents the apps source code, the security scores, and analyzed results from various tools and provides several features for the users to query this analytical dataset.

I. Introduction

Current "smartphones" are no longer really phones, they are supercomputers we carry around in our pockets which just happen to be capable of placing calls. Most smartphone vulnerabilities are not due to problems with the phone's operating system, but are caused by intentional or unintentional vulnerabilities in the phone's apps. These are created by careless developers leaving vulnerabilities in their apps or using libraries containing vulnerabilities, or by developers intentionally creating malware.

Vulnerabilities are unfortunately a pervasive aspect of software. Eliminating vulnerabilities is an important, but extremely difficult task. Software developers not only need to be concerned with vulnerabilities in their code, but with ad libraries and other 3rd party components as well [17], [18], [20]. Conversely, user's should have quality, easily accessible information to enable them to make an informed, objective decision about which apps and app versions off the best level of security. Researchers pave the way for new discoveries in both the process of developing secure software, and in understanding why vulnerabilities take place at a technical level. In order to carry out both qualitative and quantitative studies, researchers need quality and accessible data sets in which to analyze. Students need quality data, examples and exercises to educate them about the importance and process of creating secure apps.

While there are numerous tools and techniques of varying degrees of effectiveness for preventing vulnerabilities or finding them in an app currently under development, there are very few mechanisms for analyzing Android apps in the wild, the lifecycle of an app, or of understanding the Android development process.

In order to assist students, researchers, developers and general users in understanding apps, their development process, and security, we have created a free website at **http://androsec.rit.edu** which may be used to analyze Android applications.

Developers can learn from previously developed apps by seeing their mistakes both in terms of code quality and security, along with things other developers have done well. Researchers can analyze the lifecycle of apps at an individual or aggregate level. End users may use this dataset to compare apps to one another in terms of possible security vulnerabilities and code quality. They may even compare different versions of the same app against one another using these same metrics. Various app genres may also be compared in a variety of manners.

We began by collecting information on over 1,100 Android apps from F-Droid [5], which is a catalog of free an open source Android applications that contains the public version control systems of these apps. These apps ranged from small, seldom used apps; all the way to some of the more popular Android apps such as "2048", "VLC" and "Adblock Plus". We then analyzed over 4,416 different versions of these apps using several existing static analysis tools such as Stowaway [16], Androrisk [1], and Sonar [8]. We also recorded information from 435,680 version control commits including when the commit was made, who made it, and the commit message. All results are available in a publicly accessible SQLite database on our website.

In the following work, we discuss the how the website's data was created, how to use the website, and usage scenarios for classroom activities and education, researchers, developers and general users.

II. Dataset Construction

Androsec's dataset was created in two primary phases: The data collection and the static analysis phases. An overview of the collection and analysis process is shown in Figure 1.

A. Collection

In 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

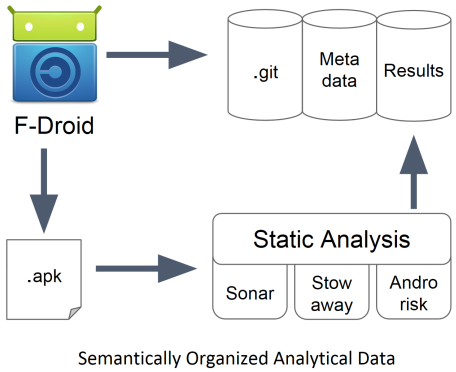

Fig. 1: Collection & Analysis Process

code of each major version, and its most recent apk (Android application) file. We collected version control information such as the committer's user name, commit time, and commit messages, ensuring all data was tagged with the version number to allow analysis over time.

B. Analysis

Once collection was complete, we continued with analysis, running a variety of static analysis tools on the app's source code. Androrisk and Stowaway [16] 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.

Stowaway is comprised of two parts - API calls made by the app are determined using a static analysis tool and the permissions needed for each API are determined using a permissions map. In this study, we use the term *over-permission* to describe a permission setting that grants more than what a developer needs for the task. Likewise, an *under-permission* is a setting for which the app could fail because it was not given the proper permissions. Over-permissions are considered security risks and under-permissions are considered quality risks.

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 [16].

Androrisk: A component of the Androguard reverse engineering tool, Androrisk determines the security risk level of an application by examining several criteria. The first set is the presence of permissions which are deemed to be more dangerous. These include the ability to access the internet, manipulate SMS messages or the ability to make a payment. The second is the presence of more dangerous sets of functionality in the app including a shared library, use of cryptographic functions, and the presence of the reflection API. The total reported security risk score for each application is recorded and available.

We chose Androrisk because it is freely available and open-source (allowing 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 [13].

III. Website

Our website has several primary components for different users. For more technical users and researchers, we have provided a SQLite database containing application, commit, and static analysis results for each application. This database can be easily downloaded for offline research. As shown in Figure 2, users may also explore the data by writing their own queries against the dataset right on the webpage.

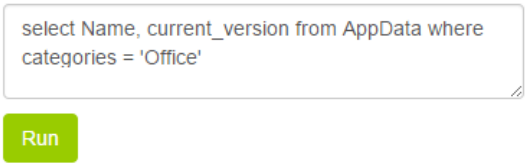

Fig. 2: Webpage Search Query

Researchers and more casual users may use various default reports available on the website. 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. As an example, we will provide information about *Wifi-Automatic* [6], a popular app with over 100,000 installs from GooglePlay. This app automatically disables a phone's wifi in order to save battery life. Figure 4 shows defect information about the app over the course of multiple versions. In this example, the first two versions of the app had two over-privileges, while the most recent version only had one, while all versions had a single recorded underprivilege.

The website also contains several built in analytics such reports showing Androrisk, over permission rate, and coding violations rate by genre. Similar information is also shown for the most popular apps.

Thomas Richards, a Senior Security Consultant for Cigital has this to say about the website:

```
''The Androsec website is an excellent
resource for developers and security
professionals to gain insight into the
increasing attack surface caused by open
source Android applications.''
```

IV. Usage Scenarios

We will next discuss several usage scenarios for the classroom, researchers, developers, and general users.


Show 10 entries Search: wifi

Name	# Versions	Current Version	Repo Type
WiGLE Wifi Wardriving	9	2.5	git
Better Wifi on/off	1	2.1.0.0	git
Open WiFi Cleaner	4	0.1.6.3	git
WiFi Automatic	9	1.4.4	git
Wifi Remote Play	2	1.12	git
Wifi Camera	1	1.6.1	git
WiFi Advanced Configuration Editor	2	0.11	git-svn
WiFiKeyboard	5	2.3.5	git
MAXS Module WifiAccess	6	0.0.1.18	srclib
Wifi Fixer	20	1.0.3.5	git

Showing 1 to 10 of 13 entries (filtered from 1,179 total entries) Previous 1 2 Next

Fig. 3: App Search for "Wifi-Automatic"

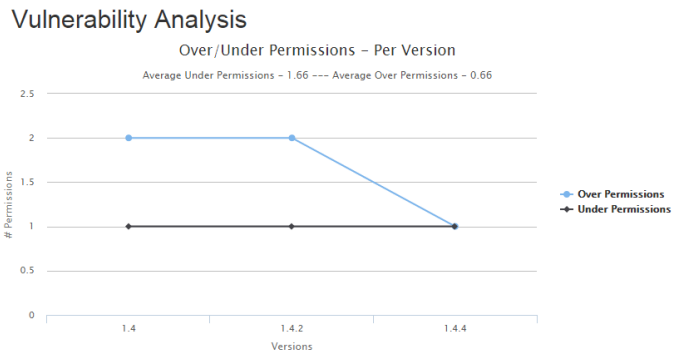

Fig. 4: Overprivileges for "Wifi-Automatic" App

A. Classroom Activities

Students and instructors may use the website and collected data for the purpose of not only learning about Android applications, but for general software development as well. Students can see and learn from the mistakes that developers have made at both the individual app level, and the aggregate level as well. While there numerous possible activities that can be conducted in the classroom using our website and data, we will discuss two which will be both valuable for instructing students on the importance and proper implementation of Android security, and will be fun and easy to implement in a classroom setting.

An example activity for a classroom could be to have a student, or group of students examine a pre-determined app, analyze its permissions issues, report on the possible negative ramifications of this permissions gap, and ultimately repair the permissions issue. In one example, students could be asked to examine "Beam File" (http://androsec.rit.edu/analytics/85), an app for transferring documents. Our analysis has shown that the first two versions of the app have 2 overprivileges and one underprivilege, while the final version has three overprivileges and no underprivileges. In the most recent version of the app, the discovered overprivileges include *NFC*, *INTERNET*, and *READ_EXTERNAL_STORAGE*. The next phase would have students report on how these overprivileges could be dangerous to a device. In this case, possible examples would include transporting sensitive information by the open internet privilege, or accessing sensitive data on the external storage device. Depending on the depth the instructor wanted the students to explore these vulnerabilities, students could also be asked to further explore vulnerabilities that have used these overprivileges in previous attacks. Students would then repair the overprivilege by eliminating these permissions from the *AndroidManifest.xml* file.

The second activity would be more research oriented. Students would use the aggregate results from the website and dataset to conduct a study on permission based vulnerabilities in Android applications. The precise details of the study would be instructor driven, but possibilities include students determining why over-privileges took place, andc which apps contained unnecessary functionality that led to avoidable vulnerabilities. Depending on the nature of the analysis, student would likely use external resources and information along with he data provided by our site. The activity would be intended for more advanced or upper level students.

B. Researcher

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 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 [9], [10], [12], [15].

We collected data from over 13,036 commits of the AndroidManifest.xml in the examined version control systems and recorded 69,707 total permissions used. This data can be used to determine how the app's settings and permission levels change and evolve over the time either at an individual app level, or a more aggregate level such as by genre.

Benchmark Dataset The collected data and the static analysis results can be used as benchmark datasets, allowing other researchers to compare their created 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.

C. Developer

There are numerous uses of this dataset for Android developers. In the event they are the creator for an app hosted on F-Droid, they may not only examine the lifecycle of their own app, but also how it compares to other existing ones as well. The majority of app developers do not use F-Droid to host their project, but they may still substantially gain from our website and data. One way to learn is by the witnessing the mistakes that others are making. Developers may observe the most pervasive over and under permissions for apps, or even genres of apps and will learn to be more vigilant in checking their own apps for these issues.

Developers often use ad libraries to make a profit from their apps. Ad libraries are given the same permissions which the app has been granted, so if the app has the ability to read sms messages, the ad library will inherit these permissions as well. Permission probing is when a 3rd party component attempts to use a permission in the hope that the attached app has requested it from the user. If the attached app has requested a permission, then the component will also have access to that permission as well. This is often done to collect, and transmit potentially sensitive information which should not be normally available to the 3rd party component. Grace et al. [18] found that more than half of all ad libraries try to probe for open permissions. This could often be the cause of an under-permission in an app since the ad library will try to use a permission which the developer did not request.

Research has also shown that users generally felt uncomfortable and may even delete applications when they did not understand why it requested a permission they deemed unnecessary [19]. Egelman et al. [14] found that approximately 25% of users were typically willing to pay a premium in order to use the same application, but with fewer permissions, while about 80% of users would be willing to allow their apps more permissions to receive targeted advertisements if it would save them .99 cents on the purchase of the app.

D. General User

General users would be able to use our site and data set in several ways. The first is that they could examine how apps that they use evolve over time. Perhaps the latest, greatest version version of an app they enjoy is actually *less* secure than a new version of the app. Users could also understand how many apps, which should contain relatively small amounts of permissions and rights, are often given too many permissions for the type of app they are. For example, a simple app such as a wallpaper app with a high Androrisk score indicates that it probably uses too many permissions and contains an unnecessary amount of functionality. This is problematic since relatively simple apps, such as wallpaper apps, often mask more sinister apps or open the door to vulnerabilities, either intentional or unintentional.

V. 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. [15] examined the effects that developer experience, and date and time of commit had on the bugginess of an application. Buse and Weimer [10] 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.

VI. 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 [7] 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.

While we have a high level of confidence in our chosen static analysis tools, static analysis tools are inherently imperfect and typically suffer from a variety of problems [11]. Some of which are that they can never demonstrate the correctness of an application, only that it could not find any problems, the need for human verification, and they often produce false positives and false negatives.

VII. Conclusion

We have presently our publicly accessible website (**http://androsec.rit.edu**) for use by students, instructors, developers, researchers and Android app users. We have described how the data set was created, its possible uses, and examples of how to use the website. We hope that our work assists in the education, discovery and understanding of Android vulnerabilities and why they occur.

References

[1] Androguard. https://code.google.com/p/androguard/, June 2015.
[2] Appannie. http://www.appannie.com, June 2015.
[3] Appbrain. http://www.appbrain.com, June 2015.
[4] Appszoom: Discovering the best android apps. http://www.appszoom.com, June 2015.

[5] F-droid. https://f-droid.org, June 2015.

[6] Github - wifi-automatic. https://github.com/j4velin/WiFi-Automatic, June 2015.

[7] Number of android applications. http://www.appbrain.com/stats/number-of-android-apps, June 2015.

[8] Sonarqub. http://www.sonarqube.org, June 2015.

[9] 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.

[10] 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.

[11] B. Chess and G. McGraw. Static analysis for security. *IEEE Security & Privacy*, (6):76–79, 2004.

[12] 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.

[13] 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.

[14] S. Egelman, A. P. Felt, and D. Wagner. Choice architecture and smartphone privacy: There's a price for that. In *In Workshop on the Economics of Information Security (WEIS)*, 2012.

[15] 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.

[16] 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.

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

[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] J. Lin, S. Amini, J. I. Hong, N. Sadeh, J. Lindqvist, and J. Zhang. Expectation and purpose: Understanding users' mental models of mobile app privacy through crowdsourcing. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, pages 501–510, New York, NY, USA, 2012. ACM.

[20] M. Sun and G. Tan. Nativeguard: Protecting android applications from third-party native libraries. In *Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless & Mobile Networks*, WiSec '14, pages 165–176, New York, NY, USA, 2014. ACM.