Acta Scientific COMPUTER SCIENCES

Volume 4 Issue 2 February 2022

Research Article

Evaluation of Static Analysis Tools for Mobile App Security

Ayush Maharjan[1,2], Nahida Sultana Chowdhury[1,2]* and Rajeev R Raje[2]

[1]*Modern Software Engineering, DMI, USA*

[2]*Indiana University Purdue University Indianapolis (IUPUI), USA*

***Corresponding Author:** Nahida Sultana Chowdhury, Software Engineer, DMI, Indianapolis, IN, USA.

Received: December 13, 2021

Published: January 18, 2022

Abstract

With the large number of Android apps available in app stores such as Google Play, it has become increasingly challenging to find the secure Apps. Therefore, it is very important for users to consider the security and privacy issues while selecting an app from any public app store. Many static analysis tools can identify security and privacy-related vulnerabilities in any mobile app code by highlighting potential flaws, often offering examples to resolve these flaws, and may even modify the code to remove the susceptibilities. This paper empirically compares three publicly available static analysis tools for Android Apps and investigates their pros and cons using the Ghera benchmark.

Keywords: Static Code Analysis; Android; Mobile App; Security; Privacy

Introduction

The types of security risks posed by mobile apps are quite different from the risks involved with desktop or Web software systems. Most of the mobile applications rely on user data and constantly communicate through the network with remote servers and devices. It is important to make sure that the data is protected within the device as well as when it is being transmitted over a communication channel such as WiFi, Bluetooth, NFC, etc. With the advancement of smartphones, people have become more dependent on such devices and apps they support. Many apps use important personal data of the users (such as their photos, location, personal messages, etc.), which makes the security of data even more important. Hence, it is utmost necessary to consider the security and privacy issues while downloading an app from any app store. Such an analysis of apps can assist users in ranking similar apps and make informed decisions before downloading an app for their usage.

There have been numerous efforts [1-3] on identifying the security issues in mobile apps. Several open source and commercial static analysis tools [4-7] are available that can detect security issues in the mobile apps. These tools are used during development to detect and fix security and privacy issues. However, one of the challenges that still needs to be considered, is that the tools themselves may not be reliable. Most of the static analysis tools are known to report false positives [8]. It is difficult to consider each such issue while running static analysis on large number of real-world apps. This limitation can be overcome by using a sample set of benchmark apps to analyze the reliability of the tools. This research uses Ghera android vulnerabilities benchmark [9] to benchmark three publicly available static analysis tools. The Ghera benchmark is chosen over other available benchmarks due several factors including the number of issues reported across different categories, the nature of the issues reported, and the format of the benchmark apps are organized. The rest of the paper is organized as follows: section 4 presents related work and background literature. Section 5 presents the benchmark analysis. Section 6 discusses experiments conducted and the results obtained. The paper concludes by highlighting the insights gained and presents directions for future work. The work described in the paper is a subset of the work reported in the master's thesis of the first author [10].

Related work and background

Benchmarking of tools

Several benchmarks, associated with vulnerabilities for Android platforms, have been developed for analyzing static code analysis tools. A few prominent benchmarks that were considered in this study are discussed below.

The Ghera Android Vulnerabilities benchmark suite [9] provides source code for a benign app, a malicious app and a secure apps for each specific benchmark. A benign app is the version of the application that exhibits the vulnerability; the malicious app is the application that exploits the vulnerability in the benign app; and the secure app is the application with the security vulnerability removed from the benign app. Each benchmark also provides a summary of the vulnerability along with the affected Android versions and description and example of the vulnerability being demonstrated through the benign and the secure apps. The Ghera benchmark has a total of 60 benchmarks categorized into seven groups - Cryptography, Inter-Component Communication (ICC), Networking, Non-API, Permission, Storage, System, and Web.

Damn Insecure and Vulnerable App (DIVA) [10] is an app that contains insecure and vulnerable code. It was originally intended as a learning tool for Android developers to understand different security vulnerabilities, but it has been used by security professionals for penetration testing. It includes various challenges such as insecure logging, hard coding issues, insecure data storage, input validation issues, access control issues, etc. [11].

Purposefully Insecure and Vulnerable Android Application (PIVAA) [12] is another insecure and vulnerable app that was designed as an improvement over the outdated DIVA.

DroidBench 14 [13] is a micro-benchmark suite designed to evaluate the effectiveness of Android taint-analysis tools. It comprises of 120 test cases for data leakage in Android apps. The test cases cover the leakages related to Java (such as Arrays and Lists, Callbacks, Reflection) and Android APIs (such as Lifecycle, Inter-App Communication, Inter-Component Communication).

ICC-Bench [14] suite is a specialized repository of benchmark apps focused towards Inter-Component data leakage in Android apps. It consists of 24 small apps representing various vulnerabilities related to Inter-Component Communication (ICC).

DialDroid-Bench [15] is another benchmark suite focused towards the Android taint-analysis tools that consists of 30 real world applications. It only consists of the apk files without any source code or vulnerability details making it difficult to put it into use for analysis of the tools.

This research chooses to use the Ghera benchmark over the other benchmark for several reasons. First, the Ghera benchmark covers a wider range of issues than other benchmarks across several categories. It covers a broader range of issues than the benchmarks such as DroidBench, ICC-Bench, DialDroid-Bench, which are focused towards the taint analysis tools. Secondly, the Ghera benchmark consists of micro-benchmarks that makes it easier to look at each issue separately. Though DIVA and PIVAA also represent categories that focus on issues not just related to the taint analysis, they contain all the errors in a single application. This makes the analysis complex because it is difficult to verify if the issue is reported correctly or not. Lastly, the Ghera benchmark also provides good documentation of the issues and a version of the benchmark app with the issue fixed. This is particularly useful because it helps to recognize the false negatives reported by the tools.

Evaluating the analysis tools

Qiu., *et al*. [16] have performed an analysis of static taint analysis tools, FlowDroid, AmanDroid, and DroidSafe. They have performed the comparison using the DroidBench and ICC-Bench Benchmark Suites. In this research, we have performed a similar analysis for generic static analysis tools that were used. The tools and benchmarks are focused toward Taint Analysis and focus mostly on ICC vulnerabilities. It does not address issues such as cryptography, insecure usage of APIs, etc.

Pauck., *et al*. [17] have also performed an empirical evaluation of the static taint analysis tools used in the research community. They have used the DroidBench [13] to perform the analysis on six different tools. They have also proposed the ReproDroid framework to perform an accurate and reproducible evaluation of the static analysis tools to overcome the differences in the evaluation techniques used by the authors of the tools. This research is focused towards taint analysis as well, and does not cover several of the vulnerabilities beside leakage of information.

A survey of android security threats and defenses was conducted by Rashidi., *et al*. [18]. Many of the threats identified are still relevant, whereas some have become outdated. There were several tools gathered in the survey, but we could not download or execute any of the tools that were surveyed. Thus, none of the tools fit the needs of this research.

Tool analysis

For the selection of tools, we focused on generic static analysis tools that cover many aspects related to the security of the tools. The taint analysis tools, which are geared only towards data leakages, were not considered in our study. Dynamic code analysis tools take a long time for analysis and may not cover all execution paths and hence, those were not selected in our study as well. Among the potential tools that we identified, JAADAS [4] results in errors during the analysis of most of the selected applications. MobSF [7], AndroBugs Framework [6], and Qark [5] were tested against various applications downloaded from Apkpure [19] and ran successfully. Thus, we have used these three tools for this study.

MobSF

MobSF provides a user interface to upload apk files and performs a detailed analysis of the apk files. It displays the analysis results in the web app, but it can also return the results in json format through the Rest API. It can also perform analysis of iOS applications.

MobSF extracts all the metadata of the apk including the name, package name, the launcher activity, minimum SDK, maximum SDK, version code and the version number. It also tracks the manifest along with all the activities, services, receiver, and providers of the app. It performs various kinds of security analyses such as checking the signer certificate, checking permissions, binary analysis, manifest analysis, code analysis, file analysis, and malware analysis. It also allows dynamic analysis of the apps.

Among the analysis that MobSF provides, the manifest analysis, and code analysis contain the results of the static analysis related to most of the security vulnerabilities. Other analysis did not provide any significant evidences that could be used in our study.

Category	Vulnerability	Qark		Androbugs		MobSF	
		B	S	B	S	B	S
Crypto	BlockCipher-ECB-InformationExposure-Lean						
	BlockCipher-NonRandomIV-InformationExposure-Lean						
	ConstantKey-ForgeryAttack-Lean					P	
	ExposedCredentials-InformationExposure-Lean			P	O		
	PBE-ConstantSalt-InformationExposure-Lean	P				P	O
ICC	DynamicRegBroadcastReceiver-UnrestrictedAccess-Lean	P	O				
	EmptyPendingIntent-PrivEscalation-Lean	P	O				
	FragmentInjection-PrivEscalation-Lean						
	HighPriority-ActivityHijack-Lean						
	ImplicitPendingIntent-IntentHijack-Lean						
	InadequatePathPermission-InformationExposure-Lean			P		P	
	IncorrectHandlingImplicitIntent-UnauthorizedAccess-Lean	P	O	P		P	
	NoValidityCheckOnBroadcastMsg-UnintendedInvocation-Lean	P	O			P	
	OrderedBroadcast-DataInjection-Lean						
	StickyBroadcast-DataInjection-Lean	P					
	TaskAffinity-ActivityHijack-Lean					P	
	TaskAffinity-LauncherActivity-Lean						
	TaskAffinity-PhisingAttack-Lean					P	
	TaskAffinityAndReparenting-PhisingAndDoSAttack-Lean						
	UnhandledException-DOS-Lean						
	UnprotectedBroadcastRecv-PrivEscalation-Lean	P	O	P		P	
	WeakChecksOnDynamicInvocation-DataInjection-Lean	P	O			P	

Networking	CheckValidity-InformationExposure-Lean	P					
	IncorrectHostNameVerification-MITM-Lean	P		P			
	InsecureSSLSocket-MITM-Lean			P	O	P	O
	InsecureSSLSocketFactory-MITM-Lean			P		P	O
	InvalidCertificateAuthority-MITM-Lean	P		P			
	OpenSocket-InformationLean-Lean						
	UnEncryptedSocketComm-MITM-Lean						
	UnpinnedCertificates-MITM-Lean						
NonAPI	MergeManifest-UnintendedBehavior-Lean						
	OutdatedLibrary-DirectoryTraversal-Lean						
Permission	UnnecessaryPerms-PrivEscalation-Lean						
	WeakPermission-UnauthorizedAccess-Lean			P		P	
Storage	ExternalStorage-DataInjection-Lean	P					
	ExternalStorage-InformationLeak-Lean	P					
	InternalStorage-DirectoryTraversal-Lean						
	InternalToExternalStorage-InformationLeak-Lean						
	SQLite-execSQL-Lean						
	SQLite-RawQuery-Lean					P	O
	SQLite-SQLInjection-Lean					P	O
System	CheckCallingOrSelfPermission-PrivilegeEscalation-Lean	P	O				
	CheckPermission-PrivliegeEscalation-Lean	P	O				
	ClipboardUse-InformationExposure-Lean					P	
	DynamicCodeLoading-CodeInjection-Lean						
	EnforceCallingOrSelfPermission-PrivilegeEscalation-Lean						
	EnforcePermission-PrivilegeEscalation						
	UniqueIDs-IdentityLeak-Lean	P		P			
Web	HttpConnection-MITM-Lean						
	JavaScriptExecution-CodeInjection-Lean	P		P			
	UnsafeIntentURLImpl-InformaitonExposure-Lean						
	WebView-CookieOverwrite-Lean						
	WebView-NoUserPermission-InformationExposure-Lean					P	O
	WebViewAllowContentAccess-UnauthorizedFileAccess-Lean	P					
	WebViewAllowFileAccess-UnauthorizedFileAccess-Lean	P	O	P	O		
	WebViewIgnoreSSLWarning-MITM-Lean			P		P	
	WebViewInterceptRequest-MITM-Lean					P	O
	WebViewLoadDataWithBaseUrl-UnauthorizedFileAccess-Lean						
	WebviewOverrideUrl-MITM-Lean						
	WebviewProceed-UnauthorizedAccess-Lean						

Table 1: Ghera Results.

AndroBugs

AndroBugs is a command line tool that can perform fast analysis on large number of applications. It is a Python program that provides various commands to perform analysis on a single apk or a set of apk files. It provides commands to display the results in command line and also outputs the results into text files. It internally uses MongoDB to store the results. We are leveraging this database directly to query the analysis results.

AndroBugs reports 51 different categories of vulnerability findings with four levels (info, warning, critical, and notice. Instead of reporting only the findings, it returns all 51 categories, and uses the info level to notify the absence of the category. The warning and critical levels represent the various types of vulnerabilities. The notice level can have a generic notice that is neither a good practice, nor a vulnerability, or it can also report good practices and vulnerabilities. After manually categorizing the notice level, we obtained a total of 38 categories as vulnerabilities, 2 categories as generic messages (that can be ignored during the analysis) and 11 categories as good practices (with the Finding codes reported by AndroBugs [6] reproduced verbatim).

Qark

Qark is a static analysis tool that detects security vulnerabilities in Android apps. It is programmed in Python and provides a command line interface. It can analyze both source code and apk files. It reports the results in html or json format. Each finding reports the name, category, line number, severity (error/vulnerability/info/warning), description, and the file where the vulnerability was detected. It is not capable of detecting good practices. Qark does not enlist the vulnerabilities it detects, so we had to manually extract it from the source code. We identified a total of 45 vulnerabilities.

Experimental Results

The analysis of the three above-mentioned tools on the Ghera benchmark is discussed in this section.

Dataset details

The Ghera benchmark, as indicated earlier, consists of 61 benchmark applications each with two versions of the application: benign(B) and secure(S). The benchmark apps and their details can be found in a bitbucket repository [20]. The benchmark applications are categorized by the different aspects of Android they affect. The Crypto category consists of vulnerabilities related to the information exposed through the ciphers and the keys used. The ICC category consists of the vulnerabilities exposed through communication between different Android components. The benchmarks related to the networking security are provided under Networking category. The Non-API category consists of vulnerabilities exposed through vulnerable and outdated libraries. The permission category relates to the vulnerabilities exposed through improper usage of permissions. The Storage category consists of the file system and database vulnerabilities. Improper usages of lower-level Android APIs are categorized under the System category. Finally, the Web category consists of vulnerabilities through usage of web through the URLs, webview and javascript code.

Benchmark analysis

For each benchmark app, a specific tool, if behaving correctly, should detect the vulnerability in the benign but not in the secure app. The results of applying the three tools on the Ghera benchmark are shown in the table 1. The P indicates the detection of vulnerabilities in the benign app and O represents the false positives detected in the secure app.

Qark detected the highest number (19) of the benign applications but also falsely identified vulnerabilities in nine of the secure application. MobSF had a similar performance, correctly identifying 18 and erroneously marking 9 applications. AndroBugs the lowest number (13) of vulnerability identifications in benign apps, but it had the just 2 false positive for secure applications. We can, from Table 2, notice that the different tools performed well in different categories of the Ghera benchmark. The Qark and MobSF had most benign application detections in the ICC category, but MobSF had a lower number of false positives for secure applications. MobSF suffers from wrongly identifying the secure applications in other categories, while Qark performs relatively well. AndroBugs is good at detecting Network, Web and ICC categories, which are all related to communication with external agents. The overall count for each tool and categories is shown in table 2.

Each benign application identified is a true positive, and the one that is not detected is a false negative. In the case of secure applications, when a vulnerability is detected, it is a false negative, otherwise it is a true negative. For the three tools, based on their performance on the Ghera benchmark, we calculated their F-scores and those are summarized in table 3.

Category	Qark		Androbugs		MobSF	
	Benign	Secure	Benign	Secure	Benign	Secure
Crypto	1	0	1	1	2	1
ICC	7	6	3	0	7	1
Networking	3	0	4	1	2	2
NonAPI	0	0	0	0	0	0
Permission	0	0	1	0	1	0
Storage	2	0	0	0	2	2
System	3	2	1	0	1	1
Web	3	1	3	1	3	2
Total	19	9	13	3	18	9

Table 2: Ghera count by category.

The F1-scores, as indicated in table 3, are 0.41 for MobSF, 0.42 for Qark, and 0.33 for AndroBugs. Hence, as seen from Tables I and III, we conclude that MobSF and Qark have a similar performance, while AndroBugs shows the worst performance on the Ghera benchmark suite.

Limitations

In this study, we have conducted an evaluation of three publicly available static code analysis tools based upon the vulnerabilities detected in the benign apps, and the false positives detected in the secure apps of the Ghera benchmark suite. The tool selection, in our study, has been limited because there are only a limited number of publicly available tools which run successfully without producing errors. Our experiments could be emulated with other static code analysis tools if and when they become freely available.

Tool	Precision	Recall	F-Score
MobSF	0.66666667	0.29508197	0.40909091
Qark	0.67857143	0.31147541	0.42696629
AndroBugs	0.8125	0.21311475	0.33766234

Table 3: Precision, Recall, and F-Score for Ghera Benchmark Results.

Though the Ghera benchmark consists of a large number of vulnerabilities, there are numerous other vulnerabilities in other areas of the Android framework (e.g., Camera, Networking) that are not covered by it. Also, newer vulnerabilities get introduced with newer Android versions. Thus, the Ghera benchmark suite can itself be extended with additional vulnerabilities that the tools may need to be tested against.

Conclusion and Future Work

An evaluation of three static code analysis tools (MobSF, Qark, and AndroBugs) against the Ghera benchmark suite is described in this paper. Our experiments indicate that MobSF and Qark had a better overall performance (as indicated by their F-scores) than AndroBugs. However, AndroBugs had higher precision and was able to detect vulnerabilities with a low number of false positives than the other two tools.

Our experiments also highlight different categories of vulnerabilities that are detected by these three tools. All tools can detect vulnerabilities in most of the categories in the Ghera benchmark suite. All three tools are also able to detect some of the common vulnerabilities, but each of them detects a few vulnerabilities that are not detected by other two tools.

This research can form a comprehensive basis while selecting tools for performing a security analysis of Android apps. Such a selection will allow a ranking of tools with appropriate weights assigned to them and will be beneficial for the users before they download any specific app on their Android device.

Acknowledgements

Ayush Maharjan was supported by the University fellowship provided by IUPUI and Nahida Sultana Chowdhury was supported

by the Department of Computer Science at IUPUI during this research. A. Maharjan and N. Chowdhury are currently employed at DMI - DMI has supported the publications charges for this article.

Bibliography

1. Z Qu., *et al*. "Dydroid: Measuring dynamic code loading and its security implications in android applications". In 2017 47th Annual EEE/IFIP International Conference on Dependable Systems and Networks (DSN) (2017): 415-426.

2. K Hamandi., *et al*. "Android SMS malware: Vulnerability and mitigation". In 2013 27th International Conference on Advanced Information Networking and Applications Workshops (2013): 1004-1009.

3. S Fahl., *et al*. "Why eve and Mallory love android: An analysis of android ssl (in) security". in Proceedings of the 2012 ACM Conference on Computer and Communications Security, ser. CCS'12. New York, NY, USA: Association for Computing Machinery (2012): 50-61.

4. "JAADAS Online". https://github.com/flankerhqd/JAADAS

5. "QARK Online". https://github.com/linkedin/qark

6. "Androbugs Framework Online". https://github.com/AndroBugs/AndroBugsFramework

7. "Mobile Security Framework Online". https://github.com/MobSF/Mobile-Security-Framework-MobSF

8. J Brittany., *et al*. "Why don't software developers use static analysis tools to findbugs?" 35th International Conference on Software Engineering (2013).

9. J Mitra and VP Ranganath. "Ghera: A repository of android app vulnerability benchmarks". in Proceedings of Promise (2017).

10. A Maharjan. "Ranking of Android Apps based on Security Evidences". MS Thesis, IUPUI (2020).

11. G Michael., *et al*. "Information-flow analysis of android applications in droid safe". in NDSS Symposium, (2015).

12. W Fengguo., *et al*. "Aman-droid: A precise and general inter-component data flow analysis framework for security vetting of android apps". *ACM Transactions on Privacy and Security* (2018).

13. "DIVA Android Online". https://github.com/payatu/diva-android.

14. "Purposefully Insecure and Vulnerable android Application". https://github.com/HTBridge/pivaa

15. "DroidBench 2.0". https://github.com/secure-software-engineering/DroidBench

16. N S Chowdhury and R R Raje. "A holistic ranking scheme for apps". 21st International Conference of Computer and Information Technology (2018).

17. N S Chowdhury and R R Raje. "Disparity between the programmatic views and the user perceptions of mobile apps". 20th International Conference of Computer and Information Technology (2017).

18. N S Chowdhury and R R Raje. "SERS: A security-related and evidence-based ranking scheme for mobile apps". IEEE International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications (2019).

19. "Apkpure". https://apkpure.com

20. J Mitra and VP Ranganath. "Ghera Android App Vulnerabilities benchmark".