Bachelor Thesis

Enhancing a Static Analysis Framework for Android Security Analysis

Jonathan-Can Kleiner

783058

B.Sc. Wirtschaftsinformatik

Potsdam, XX.09.2019

1. Supervisor	**Prof. Dr.-Ing. Christian Hammer**
2. Supervisor	**Dr.-Ing. Abhishek Tiwari**

GLIEDERUNG (Inhaltsverzeichnis)

Right align a line: Select line → Format → Paragraph → Tabs → Enter Position, click right, click dots
Same for removal of right alignment because now indentation wont work

Abkürzungs-, Abbildungs- und Tabellenverzeichnis

SUMMARY

Abstract in German

Abstract in English

What is the problem / motivation?
What do we propose, what we are presenting in this paper which solves that problem?

This work has the focus on reproducing the paper “Slicing Droids...” published in 2012? They proposed a framework which does Slicing. But since Android versions, techniques and tools are rapidly evolving, these kind of frameworks becomes obsolete quickly. Therefore we enhanced SAAF and did an updated analysis on more recent Market and Malicious Android sample set.

NO FOOTNOTES, EVERYTHING GOES TO REFERENCES

1. Introduction

The impact that smartphones and other kids of mobile devices have in our day-to-day life is undeniable. This impact is destined to become larger since the usage of mobile devices is rapidly growing. Among all mobile operating systems, Android has the largest global market share with 86%[1]. As of May 2011, 4.5 billion apps have been downloaded from the centralized application marketplace Google Play store and over 200,000 apps were published[3]. In 2016, these numbers increased to 82 billion downloaded apps and in 2017, the number of published Android apps reached over 3.5 million[2]. There are also a significant amount of unofficial third-party marketplaces where a user has access to millions of Android applications besides the official Google Play store.

Android relies on a permission-based security model where the user is asked if the application can have certain permissions, either before downloading the application or during runtime, when the app needs a permission to access a resource or execute a functionality. One downside of this model is that it might lead to an application with more permissions granted then they actually need, which is generally called a “permission gap”[4]. Another downside is, once a permission is granted, there are no other security mechanisms which prevents a malicious or vulnerable app to leak sensitive information or execute malicious instructions such as sending premium SMS messages.

This makes Android applications a perfect target for attackers with malicious intent.. According to Check Point Research, banking malware which became one of the most popular malware type in the recent years had a rise of more than 50% compared to last year[5]. These types of banking malware aims to steal credentials, transaction data or even transfer money from the victims account. Another newly discovered mobile malware is called “Agent Smith” and it already infected around 25 million devices[6]. It can replace already installed applications with the malicious versions using various Android vulnerabilities.

The number of malicious software is increasing with time and the only barrier for publishing new applications on the official Google store is the Google security system while the unofficial third-party market places usually does not have any security measures[7, 8]. Google Play uses the bouncer system, where an application will undergo security testing before it is allowed to be uploaded but even the Google Play security can be circumvented[10]. In 2017, Rahman et al. [9] analyzed 756 Google Play apps, 12% were detected as malware by at least one anti-virus tool and 2% were detected as malware by more than 10 tools.

Having these points in mind, it is crucial to develop efficient and effective ways to detect suspicious behavior patterns in mobile applications. These tools, which analyzes applications in an automated way can give us a detailed insight about the different parts of an app. This kind of tool can be then extended and developed even more to satisfy different needs of different domain requirements.

In this work, we revisited the study done by Hoffmann et al. [11]. They proposed SAAF, a static malware analysis framework for Android applications which is capable of disassembling and analyzing Android applications in an automated way. This framework analyzes smali code with the help of apktool[12]. Smali code is a disassembled version of the Dalvik Executable (DEX) file which is used by the Android's Java virtual machine implementation. This form of disassembling is chosen over Java decoders because it is more robust in nature[13].

SAAF is a static source-code analyzer with the main focus on performing backward slicing on certain parameters of a given method to find out how the data flows inside the application. This way, a more precise understanding of the parts and functionality of an application can be achieved and more importantly suspicious behavior patterns will be detected in an automated way. Other analysis techniques are also part of SAAF, such as control flow graph visualization, ad-related code identification and an Android Manifest parser.

In their study, [Hoffmann et al.(?)] (we will use they or their study for the sake of brevity???) more than 136.000 benign market apps and around 6.100 malicious apps were analyzed by SAAF. They reported bla bla

The Android platform had major updates over the years which changed the internals of the platform. This led to state-of-the-art analysis tools not being able to analyze newer applications mostly because the usage of an older version of the apktool. A recent study, published in 2018 [14] shows, between the six prominent Android analysis tools evaluated, only one of them was able to analyze applications with an API level 19 or higher which is supported by 95,3% of all Android devices according to Android Studio. All the other tools crashed during analysis because older versions of apktool failed to decompile newer versions of Android applications. This was also the case for SAAF.

A list about what things is contained in this paper. Show it as bullet points at the end of the “Introduction” section.

2. Background

How is Android assembled? What kind of components: Activities … Manifest …

Compiled to DEX, …

Static Analysis, backward slicing, dynamic analysis, taint analysis…

References – Need to be sorted alphabetically...

1. IDC 2019: Smartphone Market Share, https://www.idc.com/promo/smartphone-market-share/os (Last accessed on 1 September 2019)
2. Statista: Number of available applications in the Google Play Store from December 2009 to June 2019, https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/ (Last accessed on 1 September 2019)
3. Barra, Hugo: Android: momentum, mobile and more at Google I/O. Official Google Blog. Google. https://googleblog.blogspot.com/2011/05/android-momentum-mobile-and-more-at.html (Last accessed on 1 September 2019)
4. A. Bartel, J. Klein, Y. Le Traon, and M. Monperrus. Automaticallysecuring permission-based software by reducing the attack surface: an application to android. https://hal.archives-ouvertes.fr/hal-00726196/document (Last accessed on 1 September 2019)
5. Check Point Research: Cyber Attack Trends: 2019 Mid-Year Report, https://www.ispin.ch/fileadmin/user_upload/partner/pdf/CP-mid-year-report-2019.pdf (Last accessed on 4 September 2019)
6. Check Point Research: Agent Smith: A New Species of Mobile Malware, https://research.checkpoint.com/agent-smith-a-new-species-of-mobile-malware/ (Last accessed on 4 September 2019)
7. (Malware is increasing 1) Name: A Survey on Smartphones Security: Software Vulnerabilities, Malware, and Attacks, NO NEED URL or last accessed FOR PAPER??? https://pdfs.semanticscholar.org/57ca/94653a5d440a7d5574b8d400f4e055eea7f5.pdf (Last accessed on 4 September 2019)
8. (Malware is increasing 2) NAME: Security Threats on Mobile Devices and their Effects: Estimations for the Future https://www.researchgate.net/publication/297746368_Security_Threats_on_Mobile_Devices_and_their_Effects_Estimations_for_the_Future (Last accessed on 4 September 2019)

9. (Analysis that Gplay security can be circumvented) Name: Search Rank Fraud and Malware Detection in Google Play https://poloclub.github.io/polochau/papers/17-tkde-googleplay.pdf (Last accessed on 4 September 2019), OTHER PAPER FROM SAME GUYS: https://epubs.siam.org/doi/pdf/10.1137/1.9781611974348.12
10. (Google play security was circumwented) Jon Oberheide and Charlie Miller. Dissecting the android bouncer. Sum-merCon2012, New York, 2012. NO URL/PRESENTATION
11. (Slicing Droids) Name: Slicing Droids: Program Slicing for Smali Code https://www.syssec.ruhr-uni-bochum.de/media/emma/veroeffentlichungen/2013/11/12/slicing_droids_sac13.pdf (Last accessed on 4 September 2019)
12. Apktool: A tool for reverse engineering Android apk files, https://ibotpeaches.github.io/Apktool/ (Last accessed on 5 September 2019)
13. (Smali disassemble is more robust) name: W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri. AStudy of Android Application Security. In USENIX SecuritySymposium, 2011.
14. (do tools keep their promises) name: Do Android Taint Analysis Tools Keep Their Promises? (Last accessed on 5 September 2019)