Black Box Analysis of Android Malware Detectors

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Guruswamy Nellaivadivelu

May 2017

The Designated Project Committee Approves the Project Titled

Black Box Analysis of Android Malware Detectors

by

Guruswamy Nellaivadivelu

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

May 2017

Dr. Mark Stamp	Department of Computer Science
Dr. Robert Chun	Department of Computer Science
Dr. Thomas Austin	Department of Computer Science

ABSTRACT

Black Box Analysis of Android Malware Detectors

by Guruswamy Nellaivadivelu

One of the challenges of combating mobile malware in Android devices is attributed to code obfuscation. Malware writers obfuscate the source code of their programs by employing various techniques that attempt to hide the true intent of the program. Malware detectors use a handful of features to classify a program as a malware. If the malware detector uses a feature that is obfuscated, then the malware detector will fail to classify it as a malicious software. In this survey, we look at works of literature that discuss code obfuscation in malware and the challenges faced by malware detectors to combat the code obfuscation.

ACKNOWLEDGMENTS

I want to thank me, myself, and I.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

Introduction

Android malware proliferation is rising exponentially. In the second quarter of 2016, 3.5 million Android malware were detected [1]. This rapid increase in Android malware has placed the focus on Android security and made it imperative to develop more efficient defensive tools for combating malware. One of the challenges faced in this area is the use of code obfuscation techniques. Code obfuscation is a method of altering a source code to hide its actual purpose. There are many ways of obfuscating a source code in an Android environment. Several software applications that are available off the shelf can be used to achieve different levels of code obfuscation [2]. In order to address the problem of strengthening malware detector's strength, there are two fundamental questions that need to be addressed, as highlighted by Christodorescu et al. [3]:

1. Question 1: How resistant is a malware detector to obfuscations or variants of known malware?

2. Question 2: Can using limitations of a malware detector in handling obfuscations determine its detection algorithm?

The two questions above can be used as a way to gauge how good a malware detector will perform against obfuscated code.

1.1 Code Obfuscation

Code obfuscation is the process by which source code is manipulated to hide its true intentions. Code obfuscation is increasingly becoming a common tool to avoid detection by traditional malware detectors.

There are many different types of code obfuscation. The most basic type of code obfuscation involves the encryption of all the strings that are used in the code.

This overrides the detection mechanism of most of the traditional malware detectors. Some advanced malware detectors account for this encryption and are able to identify malware files. There are a host of other obfuscation techniques that can be employed by malware writers. Some of these include the obfuscation of function calls, permission hiding, and insertion of dead code.

1.2 Challenges

The challenges associated with code obfuscation primarily deal with the problem of maintaining the core functionality of the code, while making it difficult for malware detectors to detect their true purpose. This challenge becomes easier for malware writers when dealing with Android malware. The reason for this is associated with the permission levels of applications running on Android platform. Unlike anti-virus programs that run on computers, the Android system provides the same set of permission levels to the anti-virus application and the application that is being scanned. This is a major limitation for malware detector writers.

With the advent of more sophisticated tools for the encryption of source code for malware files, it is becoming increasingly difficult to differentiate instances where obfuscation is used for a genuine security reason and instances where it is being used with a malicious intent.

1.3 Objective

The primary objective of this project is to make malware detectors more responsive to the code obfuscation techniques employed by malware writers. The objectives can be broken down into the following two points:

1. 1.Identify the malware features that are used by a malware detector by encrypting it.

2. 2.Modify an existing malware detector to overcome the limitations of code

obfuscation.

The first step in implementation will be the identification of the factors in a malware that are taken into consideration by a malware detector. To achieve this, we will begin by encrypting various parameters of a malware and running it through a malware detector [4]. By following this approach, we can identify the exact scenario when a malware is no longer classified as a malware by our malware detector. Once we identify the features that are required by a malware detector, we will use this information to make the malware detectors process the obfuscated part of the code as well. This will make our malware detector more robust and improve their performance.

CHAPTER 2

Literature Survey

2.1 Introduction

In this chapter, we present the results of a literature survey that was performed to identify the current state of obfuscation mechanisms and their impact to the field of code obfuscation.

2.1.1 Code Obfuscation and Malware Detectors

The efficiency of malware detectors against code obfuscation has been a point of discussion amongst malware researchers for a very long time. A lot of research has been done on the robustness of malware detectors against high levels of obfuscation [3]. The issue of malware detector's strengths against obfuscated malware had been discussed as early as 1996, as can be seen in the quote by S.Gordon and R.Ford [4]:
'The evaluation of anti-virus software is not adequately covered by any existing criteria based on formal methods. The process, therefore, has been carried out by various personnel using a variety of tools and methods.'

2.1.2 Program Obfuscation

There has been a lot of theoretical research on the different aspects of obfuscation and on ways to improve it. Most of this research has been successful in arriving at a conclusion on the efficiency of the cryptographic problems of encryption, authentication and protocol [5]. But the problem of program obfuscation has remained an area within cryptography in which theoretical research has been inadequate. In their seminal paper on program obfuscation, Barak et al. [5] propose to represent program obfuscation as below: An obfuscator O is said to be an efficient compiler if it takes as input a program P and produces a program O(P) and satisfies the following two conditions:

1. Functionality: O(P) computes the same function as P

2. 'Virtual Black Box'property: Anything that can be efficiently computed from

O(P) can also be computed by P.

The paper by Christodorescu et al. [3] lists various ways to test and achieve program obfuscation in general. A detailed analysis of the various obfuscation methods is also discussed in the paper. One interesting angle explored by the paper deals with assigning mathematical equations to measure the effectiveness of the individual obfuscators. This lets us quantify the different obfuscators and rank them against each other. One of the evasion methods employed in malware obfuscation is polymorphism. It is a method by which a program evades various detection tools by mutating into different forms. In the paper by Rastogi et al. [6], the authors develop and propose a framework called ‘DroidChameleon ’that provides a way to transform Android applications into different forms with minimal user involvement.

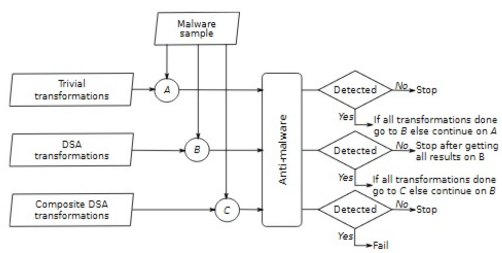

Figure 1: Evaluating anti-malware

As shown in Fig. 1, the authors apply various transformations on a malware sample dataset. The output of all these transformations are processed by a malware detector (referred here as Anti-malware). The input to the anti-malware is processed sequentially. After each transformation, the anti-malwareâĂŹs output is evaluated and if the malware detection fails, the next level of transformation is applied. This helps rank the various malware detectors against each other for accurate analysis.

2.2 OBFUSCATION IN ANDROID MALWARE

A report by Google stated that a majority of malware detectors work as a binary classifier [7]. They classify an application as a malware or a benign file. In order to effectively eliminate malicious applications, it is important that malware detectors do more than just identify malware. They should be able to isolate the core parts of the application that perform the malicious acts and work at fixing the loopholes that let the program act in a malicious way. More recent malware applications employ a variety of tricks, in addition to traditional code obfuscation mechanisms. For instance, a variant of Android malware, known as Android/BadAccent, is a known banking Trojan, that steals credentials used in banking applications [8]. A variant of this malware used a mechanism known as âĂIJTapjackingâĂİ to extract the credentials from the users. In this form of attack, a screen is displayed to the user, while a second screen is hidden behind the actual visible display [9]. When a user clicks a button on the screen, assuming it to be the one that is displayed, the underlying screen gathers the input and processes the command. This is a common method of gathering details from unsuspecting users.

2.2.1 STATISTICAL ANALYSIS TECHNIQUES AND ANDROID MALWARE

One widely used approach for analyzing malware samples is the usage of statistical methods. In such methods, the Android executable file (with the extension apk), is decompiled to get the original source code. Due to the Android operating system being written in Java, it is easy to reverse engineer an apk file to retrieve the source code. This opens up many opportunities for performing statistical analysis on the obtained raw data. This also lets a researcher perform various operations on the source code, and then repackage it back into an apk. In the approach known as AndroSimilar, Faruki et al. [10] propose a new algorithm that takes into consideration various features that are known to be present in malware alone. The AndroSimiar approach, as shown in Fig. 2 decompiles an apk file and repackages it after feature extraction. To extract the features, the algorithm incorporates apps from the Google Playstore and other third party applications. These features are normalized and fed into a signature generation engine, that provides a unique signature for each malware. This is used as reference for detecting future malware applications.

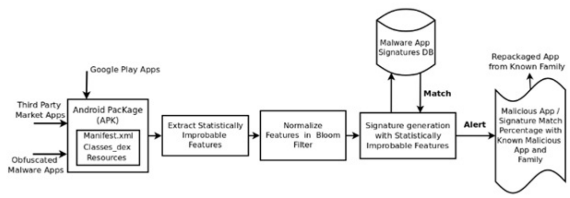

Figure 2: AndroSimilar

2.3 Conclusion

Malware in mobile devices is no longer a problem confined to labs and research areas. The rapid increase in access to computers has helped malware writers create specific, targeted programs that perform with high efficiency and exploit vulnerabilities in different operating systems. The amount of research being done in malware analysis and, more specifically, in Android malware, is in the right direction. In the fight against sophisticated metamorphic malware, it is imperative that the malware detector is better than the malware creator. In this paper, we have explored various work, that dealt with the different aspects of malware obfuscation and ways to overcome the shortcomings in today's version of malware detectors. The future of malware looks very bright and it is hoped that the malware detectors of the future will be up to the task at hand.

CHAPTER 3

The Files

For this thesis format, you LaTeX the file `thesis.tex`. But first, you need to make some modifications to `thesis.tex`. The title of your report, committee members, etc., are specified in `thesis.tex`. All of the things that you need to modify are indicated by comments beginning with five consecutive asterisks, so search for "*****" in `thesis.tex` and make the necessary changes.

You put the actual content of your report in the following files:

- `abs.tex` --- abstract
- `ack.tex` --- acknowledgements
- `chap1.tex`, `chap2.tex`, and so on --- chapters
- `bib.tex` --- references
- `appA.tex`, `appB.tex`, and so on --- appendices (if any)

CHAPTER 4

Cats

I do not usually talk about myself, but when I do, I talk about my cat.

In April 2010, the American economy was gripped by its most severe economic crisis in at least a generation. There was in a deep recession. It was in this climate that a great savior was born; her name is Muffins.

Figure 3: A Week Old Muffins

She is the first cat I have ever owned. She is a purebred Siberian. Her interests include:

- Sleeping
- Grooming herself
- Twister

The reason I chose to get a Siberian cat is because they are purported (perhaps dubiously) to be hypoallergenic. They are also a very healthy breed as they were more recently domesticated.

4.1 Career Goals

I am impressed by the ground working done by the renowned dog rehabilitator Cesar Millan, host of the critically acclaimed television show "Dog Whisperer." It is my hope that I could have an equally successful television program which I will entail "Cat Telepathist" where I will rehabilitate troubled and stray cats through the power of extrasensory perception (ESP).

If my "Cat Telepathist" show were not to find a niche with audiences, my second television show is modeled after the former television show "Crossing Over with John Edward." [?] I plan to call the show "You have Cat to be Kitten Me with Zayd Hammoudeh" and will be used to give grieving families the opportunity to communicate with their deceased feline family members.

Other TV show ideas I would like to pursue include:

- Cat Mustache Aficionado - A daily discussion of cat whiskers.
- Alien vs. Predator vs. Cat - It is like the movie "Alien vs. Predator" but instead of it being a two party battle, it is a triple threat match with a cat.
- Cat Hoarders - A show where we do an intervention with a cat that hoards some type of item.

LIST OF REFERENCES

[1] M. A. D. F. D.Emm, R.Unuchek, ‘‘It threat evolution in q2 2016,’’ 2016, accessed 2016-10-10.

[2] A.Apvrille and R.Nigam, ‘‘Obfuscation in android malware, and how to fight back,’’ *Virus Bulletin*, pp. 1--10, 2014.

[3] M.Christodorescu and S.Jha, ‘‘Testing malware detectors,’’ in *ACM SIGSOFT international symposium on Software testing and analysis*, New York, NY, USA, 2004, pp. 34--44.

[4] S. Gordon and R. Ford, ‘‘Real world anti-virus product reviews and evaluations--the current state of affairs,’’ in *Proceedings of the 1996 National Information Systems Security Conference*, 1996.

[5] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang, ‘‘On the (im) possibility of obfuscating programs,’’ in *Annual International Cryptology Conference*. Springer, 2001, pp. 1--18.

[6] V. Rastogi, Y. Chen, and X. Jiang, ‘‘Droidchameleon: evaluating android anti-malware against transformation attacks,’’ in *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*. ACM, 2013, pp. 329--334.

[7] Google, ‘‘Android security 2014 year in review,’’ https://static.googleusercontent.com/media/source.android.com/en//security/reports/Google_Android_Security_2014_Report_Final.pdf, accessed 2016-12-01.

[8] S. Rasthofer, I. Asrar, S. Huber, and E. Bodden, ‘‘An investigation of the android/badaccents malware which exploits a new android tapjacking attack,’’ Technical report, TU Darmstadt, Fraunhofer SIT and McAfee Mobile Research, Tech. Rep., 2015.

[9] Q. A. Chen, Z. Qian, and Z. M. Mao, ‘‘Peeking into your app without actually seeing it: Ui state inference and novel android attacks,’’ in *23rd USENIX Security Symposium (USENIX Security 14)*, 2014, pp. 1037--1052.

[10] P. Faruki, V. Ganmoor, V. Laxmi, M. S. Gaur, and A. Bharmal, ‘‘Androsimilar: robust statistical feature signature for android malware detection,’’ in *Proceedings of the 6th International Conference on Security of Information and Networks*. ACM, 2013, pp. 152--159.

APPENDIX A

Zorak Likes Beans

A.1 Oh Yes He Does

Appendices can have sections and subsections and so on.

A.2 Really

Sections, subsections, or whatever should come in pairs.

APPENDIX B

Everybody Wants to Be Space Ghost

Space Ghost:	Everybody wants to be Space Ghost
	Everybody near and far
	Hey, ma, look at me, I'm on TV
	Everybody wants to be a star
	I'm Space Ghost, Mr. Space Ghost
	I've got big muscles, And I can dance
	When Zorak tries to bug me, I zap him with my power bands
Zorak:	Uhhhh... I don't think that will be necessary
Space Ghost:	I think I'll zap him with my power bands
	On Saturn and Jupiter and Neptune, too
	I've been hearing it from coast to coast
	I'd give anything, If for just one day
	I could be a super hero like Space Ghost
Zorak:	Ugghh... Why would anyone want to be Space Ghost?
Space Ghost:	Everybody wants to be just like me!