

UNIVERSITY OF TECHNOLOGY SYDNEY

FACULTY OF ENGINEERING AND INFORMATION TECHNOLOGY

Machine Learning Detection and Analysis on Obfuscated Android Malware

by

Yanxin Zhang

A Thesis Submitted

**for the Degree of
Doctor of Philosophy**

Sydney, Australia

January 20, 2022

Certificate of Authorship/Originality

I, Yanxin Zhang declare that this thesis is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the School of Computer Science, Faculty of Engineering and IT at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Production Note:

Signature: Signature removed prior to publication.

Date: January 20, 2022

Abstract

In recent years, the Android platform is the prevalent operating system platform, and it accounts for more than half of the world's mobile operating system market share. With the popularity of Android smartphones and Android tablets, Android-based malware has also developed rapidly, so malicious code detection technology based on the Android platform has been proposed, and Android application security must be carefully studied and have an extensive discussion. Meanwhile, machine learning methods can simulate the behavior of Android applications and distinguish between benign and malicious software; therefore, it is becoming the key to the effective detection of Android malware. However, malware developers may understand detection systems and take countermeasures against detection methods in the real world. Most current research work ignores this problem.

Recent research in machine learning has shown that learning-based systems are susceptible to well-designed adversarial examples. For instance, malware detection based on static analysis will encounter the challenge of code obfuscation, which is the countermeasure for malware authors to bypass detection. Therefore, there is a strong demand for research on Android malware detection from the perspective of defenders and attackers.

My research direction is to explore machine learning methods to ensure the security and stability of Android applications. Malware writers often use obfuscation techniques to obfuscate Android malware to evade detection by Android malware detectors. Knowing about the impact of code obfuscation on Android malware can provide valuable insights into obfuscated malware and, therefore, the design of resilient Android malware detectors.

First, labeling malware or malware clustering is essential for identifying new security threats, triaging, and building reference datasets. The state-of-the-art Android malware clustering approaches rely heavily on the raw labels from commercial AntiVirus (AV) vendors, which causes misclustering for a substantial number of

weakly-labeled malware due to the inconsistent, incomplete, and overly generic labels reported by these closed-source AV engines, whose capabilities vary greatly and whose internal mechanisms are opaque (i.e., intermediate detection results are unavailable for clustering). The raw labels are thus often used as the only significant source of information for clustering. To address the limitations of the existing approaches, we present ANDRE, a new Android Hybrid Representation Learning approach to clustering weakly-labeled Android malware by preserving heterogeneous information from multiple sources (including the results of static code analysis, the meta-information of an app, and the raw-labels of the AV vendors) to learn a hybrid representation for accurate clustering jointly. The learned representation is then fed into our outlier-aware clustering to partition the weakly-labeled malware into known and unknown families. The malware whose malicious behaviors are close to those of the existing families on the network is classified using a three-layer Deep Neural Network (DNN). The unknown malware is clustered using a standard density-based clustering algorithm. The evaluation shows that ANDRE effectively clusters weakly-labeled malware that cannot be clustered by the state-of-the-art approaches, achieving comparable accuracy with those approaches for clustering ground-truth samples.

Second, Android piggybacked malware (i.e., apps that piggyback malicious code) are becoming ubiquitous in app stores. Malware writers often use obfuscation techniques to obfuscate piggybacked apps to evade detection by Android malware detectors. Previous studies in this field have focused on the impact of code obfuscations on the detection of piggybacked malware, but the impact of code deobfuscation on detecting obfuscated piggybacked apps has rarely been investigated. Knowing about the impact of code deobfuscation can provide helpful insights into obfuscated piggybacked apps and, therefore, the design of resilient Android malware detectors. We conduct an empirical study of code deobfuscations on detecting obfuscated Android piggybacked apps, focusing on three types of malware detectors: commercial anti-malware products, machine learning-based detectors, and similarity-based detectors. We observe that code deobfuscations can have various impact on the detection rate for different types of malware detectors, such as similarity-based or

machine learning-based detectors. Also, we observe that the examined deobfuscation tools have a different impact on obfuscated piggybacked apps after deobfuscations.

Third, Android malware adversarial samples can help test the weaknesses of antivirus products to improve their robustness against emerging and sophisticated Android malware. A typical method to generate testing samples is to transform a single Android malware into a range of samples through code obfuscation. Existing obfuscation tools often have many obfuscation options, which can generate a large number of testing samples. However, accurately identifying top evasive samples to gain an optimal cost-benefit trade-off is a challenge. To address the above issue in adversarial malware generation for robustness testing of antivirus products, we present ALBS(Active Learning-based Sampling), an active-learning-based approach, by constructing an adversarial ranking model to select the most powerful evasive adversarial samples. First, ALBS generates all Android malware adversarial samples and then extracts the adversarial samples' representation using both unstructured and structural code features. Then, it utilizes the multi-view multi-learner (MVML) active learning method to train a model which ranks the adversarial samples using multiple learning-to-rank (LTR) algorithms. Our empirical experiments demonstrate the effectiveness of ALBS for generating adversarial samples for antivirus product robustness testing. The experiment result indicates that the proposed active learning approach can optimize the adversarial sample generation process and effectively select the most potent testing samples, improving the efficiency of Android antivirus products' robustness testing.

Acknowledgements

I want to express my deep gratitude to all those who have given me help and support in writing my thesis. First of all, I would like to thank my supervisors, Dr. Yulei Sui and Prof. Ivor Tsang, for their guidance, support, and advice during my PhD study. They discovered my potential, encouraged me to go further, and provided precious lessons in many different fields. Their meticulous academic attitude and dedicated work style have always been my role models. Also, I am grateful to the University of Technology Sydney for providing generous supports and attractive academic environment during my candidature career.

I would also like to thank the colleges in my group for discussing research work with me and sharing their knowledge and suggestions generously. They provided many insightful discussions for my research and offered various help to my personal life.

Finally, I would like to thank my parents and wife for their love, care, and continuous support and encouragement. Without their valuable support, it would be impossible for me to finish this thesis.

Sydney, Australia, January 20, 2022

List of Publications

Below is a list of publications that are included in this thesis as Chapter 3, 4, 5 respectively.

Chapter 3:

Zhang, Y., Sui, Y., Pan, S., Zheng, Z., Ning, B., Tsang, I. and Zhou, W., 2019. "Familial clustering for weakly-labeled android malware using hybrid representation learning." *IEEE Transactions on Information Forensics and Security*, 15, pp.3401-3414. Status: Published

Chapter 4:

Zhang, Y., Xiao, G., Zheng, Z., Zhu, T., Tsang, I.W. and Sui, Y., 2020, December. "An Empirical Study of Code Deobfuscations on Detecting Obfuscated Android Piggybacked Apps." *In 2020 27th Asia-Pacific Software Engineering Conference (APSEC) (pp. 41-50). IEEE*. Status: Published

Chapter 5:

Zhang, Y., Cai, H., Tsang, I.W. and Sui, Y. "ALR: Active Learning Based Ranking for Adversarial Android Malware Generation." *Submitted to IEEE Transactions on Reliability*. Status: Under review

Other publications during candidature:

Zhang, Y.. "Obfuscation Resilient Lightweight Android Repackaged Apps detection." *Submitted to IEEE Transactions on Dependable and Secure Computing*. Status: Under review

Contents

Certificate	1
Abstract	1
Acknowledgements	1
List of Publications	1
1 Introduction	14
1.1 Background	14
1.1.1 Weakly-labeled Android Malware Family Clustering	14
1.1.2 Investigating Code Deobfuscations on Detecting Obfuscated	
Piggybacked Apps	16
1.1.3 Active Learning Based Ranking for Adversarial Android Mal-	
ware	18
1.2 Research Topics	19
1.2.1 Weakly-labeled Android Malware Family Clustering	19
1.2.2 Investigating Code Deobfuscations on Detecting Obfuscated	
Piggybacked Apps	20
1.2.3 Active Learning Based Ranking for Adversarial Android Mal-	
ware	20

1.3	Research Contributions	21
1.3.1	Weakly-labeled Android Malware Family Clustering	21
1.3.2	Investigating Code Deobfuscations on Detecting Obfuscated Piggybacked Apps	22
1.3.3	Active Learning Based Ranking for Adversarial Android Malware	22
1.4	Thesis Organization	23
2	Literature review	24
2.1	Weakly-labeled Android Malware Family Clustering	24
2.1.1	Android Malware Detection	24
2.1.2	Android Malware Classification and Clustering	25
2.1.3	Representation Learning	26
2.2	Investigating Code Deobfuscations on Detecting Obfuscated Piggybacked Apps	26
2.2.1	Android Obfuscation Strategies	26
2.2.2	Android Deobfuscation	29
2.3	Active Learning Based Ranking for Adversarial Android Malware	30
2.3.1	Active Learning	30
2.3.2	Android Antivirus Products Penetration Testing	31
3	Weakly-labeled Android Malware Family Clustering	32
3.1	Introduction	32
3.2	Framework Overview	37
3.2.1	Feature Extraction	38
3.2.2	Hybrid Representation Learning	38
3.2.3	Outlier-aware Weakly-labeled Malware Clustering	39
3.3	Our Approach	40
3.3.1	Android Malware Feature Extraction	40
3.3.2	Android Network Representation Learning (ANDRE)	43
3.3.3	Model Details	44
3.3.4	Outlier-aware Android Malware Clustering	49

3.4	Evaluation	50
3.4.1	Experimental Setup and Implementation	50
3.4.2	Evaluation Methodology	51
3.4.3	Comparing with State-of-the-art Tools and Different Baseline	
	Settings	52
3.4.4	Result Analysis for Weakly-labeled Malware	56
3.4.5	Case Studies for Weakly-labeled Malware in Inlier	57
3.4.6	Case Studies for Weakly-labeled Malware in Outlier	58
3.5	Discussions and Limitations	59
4	Investigating Code Deobfuscations on Detecting Obfuscated Piggybacked Apps	63
4.1	Introduction	63
4.2	Research Methodology	67
4.2.1	Dataset and App Generation	67
4.2.2	Targeted Systems	69
4.2.3	Experiment Designs	74
4.3	Results and Analysis	74
4.3.1	RQ1. Impact of Code Obfuscations	75
4.3.2	RQ2. Impact of Code Deobfuscations	78
4.3.3	RQ3. Impact of Deobfuscation Tools	82
4.4	Case Studies	84
4.5	Discussions and Limitations	86
5	Active Learning Based Ranking for Adversarial Android Malware	87
5.1	Introduction	87
5.2	Our Approach	90
5.2.1	MVML Active Learning	90
5.2.2	LTR Models Synthesizing	93
5.3	Data Collection and Aggregation	96
5.3.1	Adversarial Sample Generation	96

5.3.2	Static and CFG Feature Extraction	96
5.4	Evaluation	99
5.4.1	Experimental Setup and Implementation	99
5.4.2	Evaluation Metric	100
5.4.3	Result Analysis	103
5.5	Threat To Validity	108
5.5.1	Threats to External Validity	108
5.5.2	Threats to Internal Validity	108
5.5.3	Threats to Construct Validity	109
6	Conclusion	110
6.1	Weakly-labeled Android Malware Family Clustering	110
6.2	Investigating Code Deobfuscations on Detecting Obfuscated Piggy-backed Apps	110
6.3	Active Learning Based Ranking for Adversarial Android Malware	111
7	Research Plan and Ethical Issues	112
7.1	Research Plan	112
7.1.1	VirusTotal's Labelling Policy	112
7.1.2	Active learning strategies with good versatility	113
7.2	Ethical Issues	113
	Bibliography	127

List of Figures

3.1	An example of malware with empty label.	35
3.2	An example of malware with controversial labels.	37
3.3	The overview of ANDRE.	45
3.4	The DEEPWALK (Skip-Gram) method vs our proposed hybrid learning method.	48
3.5	Performance comparison with respect to different learning models.	53
3.6	Performance comparison (permissions vs all meta-info).	55
3.7	Weakly-labeled malware detected as outliers (unknown families in the network).	60
3.8	Weakly-labeled malware detected as inliers (known families in the network).	61
4.1	Illustration of Android piggybacked malware.	64
4.2	Overview of our empirical study.	66
4.3	Example of VirusTotal.	71
4.4	The impact of code deobfuscation tools on different detectors. (a) VirusTotal, (b) Drebin, (c) CSBD, (d) Androguard, (e) SimiDroid-C, and (f) SimiDroid-R.	82
4.5	The impact of code deobfuscation tools on different types of obfuscation strategies. (a) CFF, (b) IJC, (c) SO, and (d) IO.	83

4.6 Case study: insert junkcode.	84
4.7 Case study: identifier renaming.	85
4.8 Case study: string obfuscation.	86
5.1 Number of generating 100 malware's adversarial samples given # of obfuscation options.	88
5.2 Time cost to process apps by VirusTotal.	89
5.3 The overview of ALR.	90
5.4 An example of adversarial samples on VirusTotal.	91
5.5 Recall results of ranking adversarial samples.	104

List of Tables

3.1 Malware with empty labels	33
3.2 Malware with controversial labels	33
3.3 Meta-data information extracted from an Android app	47
3.4 Comparison with AVCLASS [1] and EUPHONY [2]. The accuracy, F1 and recall data are directly from their papers	52
3.5 Comparison with different classifiers	54
3.6 Performance comparison regarding different dimensions K in the rep- resentation space	55
3.7 Results of outliers and inliers	56
3.8 Suspicious permissions used by apps in Outlier-1	58
4.1 Obfuscation Strategies	68
4.2 Number of Obfuscated Apps	69
4.3 Number of Deobfuscated Apps (Simplify)	69
4.4 Precision of Detecting Obfuscated Piggybacked Apps by Different Detectors	73
4.5 Precision of Detecting Deobfuscated Piggybacked Apps by Different Detectors (Simplify)	77
4.6 Precision of Detecting Deobfuscated Piggybacked Apps by Different Detectors (Deguard)	77

5.1	Obfuscation Strategies	93
5.2	Adversarial samples for evaluation	96
5.3	NDCG score of using Random Static method	101
5.4	NDCG score of using MVML Static method	101
5.5	NDCG score of using Random CFG method	102
5.6	NDCG score of using MVML CFG method	102
5.7	Weights of each learner	102
5.8	Top 10 strategies in the testing set ranked by ALR	107
5.9	Bottom 10 adversarial samples in the testing set ranked by ALR	107