Computers and Electrical Engineering 000 (2017) 1–9

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Machine learning aided Android malware classification[☆]

Nikola Milosevic[a], Ali Dehghantanha[b], Kim-Kwang Raymond Choo[c,*]

[a] School of Computer Science, University of Manchester, UK
[b] School of Computing, Science and Engineering, University of Salford, UK
[c] Department of Information Systems and Cyber Security, The University of Texas at San Antonio, San Antonio, TX 78249-0631, USA

ARTICLE INFO

Article history:
Received 9 November 2016
Revised 10 February 2017
Accepted 13 February 2017
Available online xxx

MSC:
00-01
99-00

Keywords:
Static malware analysis
OWASP
Seraphimdroid Android app
OWASP Seraphimdroid Android app
Machine learning

ABSTRACT

The widespread adoption of Android devices and their capability to access significant private and confidential information have resulted in these devices being targeted by malware developers. Existing Android malware analysis techniques can be broadly categorized into static and dynamic analysis. In this paper, we present two machine learning aided approaches for static analysis of Android malware. The first approach is based on permissions and the other is based on source code analysis utilizing a bag-of-words representation model. Our permission-based model is computationally inexpensive, and is implemented as the feature of OWASP Seraphimdroid Android app that can be obtained from Google Play Store. Our evaluations of both approaches indicate an F-score of 95.1% and F-measure of 89% for the source code-based classification and permission-based classification models, respectively.

1. Introduction

In our increasingly connected society, the number and range of mobile devices continue to increase. It is estimated that there will be approximately 6.1 billion mobile device users by 2020 [1]. The wealth of private information that is stored on or can be accessed via these devices made them an attractive target for cyber criminals [2]. Studies have also revealed that users generally do not install anti-virus or anti-malware app installed on their mobile devices, although the effectiveness of such apps is also unclear or debatable [3]. Hence, mobile devices are perceived by security professionals among the ”weakest links” in enterprise security.

While all mobile operating systems/platforms have been targeted by malware developers, the trend is generally to focus on mobile operating systems with a larger market share. A bigger market share [4] along with Google's flexible publishing policy on Android's official application (also referred to as app) market – Google Play – resulted in Android users being a popular target for malware developers. It is also known that Android permission-based security model provides little protection as most users generally grant apps requested permissions [5]. There have also been instances where malicious apps were successfully uploaded to Google Play [6]. This suggests a need for more efficient Android malware analysis tools.

☆ Reviews processed and recommended for publication to the Editor-in-Chief by Guest Editor Dr. A. K. Sangaiah.
* Corresponding author.
E-mail addresses: nikola.milosevic@manchester.ac.uk (N. Milosevic), a.dehghantanha@salford.ac.uk (A. Dehghantanha), raymond.choo@fulbrightmail.org (K.-K.R. Choo).

http://dx.doi.org/10.1016/j.compeleceng.2017.02.013
0045-7906/© 2017 Elsevier Ltd. All rights reserved.

Existing approaches for malware analysis can be broadly categorized into dynamic malware analysis and static malware analysis. In static analysis, one reviews and inspects the source code and binaries in order to find suspicious patterns. Dynamic analysis (behavioral-based analysis) involves the execution of the analyzed software in an isolated environment while monitoring and tracing its behavior [7].

Early approaches to mobile malware detection were based on the detection of anomalies in battery consumption [8]. Operating system events, such as API calls, Input/Output requests, and resource locks, have also been used in dynamic malware detection approaches. For example, TaintDroid is a malware detection system based on anomalies in the app's data usage behavior [9]. In [10], the authors created a system that monitored anomalies in Android Dalvik op-codes frequencies to detect malicious apps. Several approaches utilized machine learning to classify malware based on their behaviors. For example, the authors in [11] focused on run-time behavior and classified Android malware into the malware families using inter-process communications in combination with SVM. A random forest-based approach with set of 42 vectors including battery, CPU and memory usage as well as network behavior was also used for Android mawlare detection in [12]. In [13], the authors used system calls and regular expressions to detect data leakage, exploit execution and destructive apps.

In order to avoid degrading of mobile device's performance, solutions based on distributed computing and collaborative analysis for both static and dynamic malware analysis have also been proposed [7]. For example, M0Droid is an Android anti-malware solution that analyzes system calls of Android apps on the server and creates signatures which are pushed to the user devices for threat detection [14].

Static malware analysis techniques mainly rely on manual human analysis, which limits the speed and scalability of investigation. Different approaches to automate the static analysis process have also been proposed. Mercaldo et al. [15] suggested transforming malware source code into Calculus for Communicating Systems (CCS) statements and utilized formal methods for checking the software's behavior. However, their approach requires human analysts to formally describe the unwanted behavior, which could still be time-consuming. The authors in [16] proposed a methodology to generate fingerprints of apps that captures binary and structural characteristics of the app. Machine learning techniques can be used to automate static malware analysis process. In [17], pattern recognition techniques are used to detect malware, while other works used standard machine learning algorithms such as perception, SVM, locality sensitive hashing and decision trees to assist in malware analysis (see [18]). In [19], the authors extracted network access function calls, process execution, string manipulation, file manipulation and information reading, prior to applying different machine learning algorithms to classify malicious programs. In [20], the authors extracted 100 features based on API calls, permissions, intents and related strings of different Android apps and applied Eigen space analysis to detect malicious programs. Sahs and Khan used Androguard to obtain permissions and control flow graphs of Android apps and created a SVM-based machine learning model to classify Android malware [21].

In this paper, we demonstrate the utility of employing machine learning techniques in static analysis of Android malware. Specifically, techniques such as manifest analysis and code analysis are utilized to detect malicious Android apps. The contributions of this paper are two-folded:

1. We present a machine learning model for Android malware detection based on app permissions. This approach is lightweight and computationally inexpensive, and can be deployed on a wide range of mobile devices.
2. We then present a new approach to perform code analysis using machine learning, which provides higher accuracy and is capable of revealing more granular app behaviors. Static code analysis of malware is a task generally undertaken by forensics and malware analysts. However, our research results indicate the potential to automate several aspects of the static code analysis, such as detecting malicious behavior within the code.

The structure of this paper is as follows. In the next section, we present the research methodology used in this paper. Research results are then presented, followed by a discussion of the findings. Finally, the paper is concluded and several future directions are suggested.

2. Methodology

Combination of permissions may give a clear indication about the capabilities of the analyzed app(s). From combining the permissions, it can be induced weather the app may cause harm or behave maliciously. We hypothesize that malicious apps will have certain patterns and common permission combinations, which can be learned by a machine learning algorithm. On the other hand, the app code reflects the app's behavior and, therefore, is a common choice for static malware analysis. We utilize two machine learning techniques, namely: classification and clustering. As apps can be classified into malware and goodware, the task of malware detection can be modeled as a classification problem.

Classification is a supervised machine learning technique, which can be used to identify category or sub-population of a new observation based on labeled data. Clustering is an unsupervised machine learning technique that is capable of forming clusters of similar entities. Clustering algorithms are useful when only a small portion of dataset is labeled. The labeled examples can be utilized to infer the class of unlabeled data. Labels obtained through clustering can be subsequently used to retrain a classification model with more data.

Also, in this research, we conducted four experiments, namely: permission-based clustering, permission-based classification, source code-based clustering, and source code-based classification. For the training and testing of our machine learning models, we utilize M0Droid dataset, which contains 200 malicious and 200 benign Android apps [14].