O.S android-malware

Hillel Ohayon 325300820
Tom Koltunov 323929190

1 Introduction

Malware attacks are any software intentionally designed to cause disruption to a computer, server, client, or computer network, leak private information, gain unauthorized access to information or systems, deprive access to information, or which unknowingly interferes with the user's computer security and privacy.
This paper represent an innovative machine learning-based malware detection approach.

2 Name of paper

"Empirical assessment of machine learning-based malware detectors for Android".
(Measuring the gap between in-the-lab and in-the-wild validation scenarios) Kevin Allix, Tegawende F., Bissyand ´ e´, Quentin Jerome ´, Jacques Klein, Radu State. Yves Le Traon
Published online: 24 December 2014

3 feature sets

The paper proposes a feature set for machine-learning classifiers for malware detection. Machine learning-based malware detection relies on a training data that is analyzed to learn what could suggest that a given application is a potential malware.
To that end, the learning algorithm must be "told" what features are relevant in each piece of data of the dataset. Indeed, Machine Learning algorithms cannot work directly on Android applications.
Each application must be represented with an ordered list of properties-called a Feature vector in the context of Machine Learning.

4 classifier type

Generally, we analyze static classifier: Features are often extracted from program metadata or program code (binaries, bytecode, source code).
In the case of the Android Operating System, features can be extracted from application bytecode using static analysis. Indeed, Android applications are distributed in the form of .apk files which are packages containing the application's Dalvik (a virtual machine that is included in the Android OS) bytecode, assets such as images, and metadata specific to the Android platform. Android applications are generally written in Java.
The program is then compiled to Java bytecode which is converted into Dalvik bytecode. Unlike the typical binary code, Dalvik bytecode retains most of the information contained in Java bytecode. Thus, such code can be fed to Static Analysis tools that support Dalvik bytecode or after converting it back to Java Bytecode for which many analyzers exist.
In our paper's work, the static analysis will be performed using AndroGuard. We perform static analysis of Android applications' bytecode to extract a representation of the program control-flow graph (CFG).

5 GIT

https://github.com/HILLELOH/csbd-O.S-android