Journal of Information Assurance and Security
ISSN 1554-1010 Volume 8 (2013) pp. XXX-XXX
© MIR Labs, www.mirlabs.net/jias/index.html

Reverse Engineering for potential Malware detection: Android APK Smali to Java

Girish Sharma[1], Mehul Mahrishi[2] , Kamal Kant Hiran[3] and Dr. Ruchi Doshi[4]

[1]Department of Computer Science & Engineering,
Swami Keshvanand Institute of Technology, Management & Gramothan, Jaipur -302017 India
girish@skit.ac.in

[2]Department of Computer Science & Engineering,
Swami Keshvanand Institute of Technology, Management & Gramothan, Jaipur -302017 India
kamalhiran@gmail.com

[3]Department of Computer Science,
Sir Padampat Singhania University, Udaipur, India

Abstract: **Emerge of Smartphone technology has changed the way of communication and processing the data. These smart phones can perform peculiar thing which was only limited to calling and texting previously. This work presents the reverse engineering of the Android application which is the one of the most prominent Smartphone technology based on the Linux kernel. Since it is very difficult to analyze the applications by using intermediate codes like smali, jimple or bytecode, this approach can be useful for the reseachers who work on control and data flow analysis of apps.**

The objective of this work is twofold. One is to identify the components specified by the developer using the Android application's Manifest file and also those class files which have not been specified in the Manifest file . The second objective is to reverse engineer all the components and classes i.e. to convert them in respective Java code.

Keywords: Reverse Engineering, Android, Smali, Jimple, control flow, data flow

I. Introduction

Since the inception of android based systems, is one of the ubiquitous technologies which are used for sharing the resources [1]. At the same time these systems can be exploited by the malicious program writers to exploit the data or resources intentionally. As these systems are increasing exponentially the malicious writers misuse the private data of the user to fulfill their malicious activities. Another big possibility is that one application may use another apps data and could do malicious things [2].

Lot of techniques has been developed for the analysis of an android application which can provide the flow analysis of the application [3]. For example SOOT based framework "Flowdroid" is one of the tool which can be used for static taint analysis [4] which may provide good precision and recall. Flowdroid uses Jimple intermediate code to perform the analysis of the applications [5].

II. Android: Impact, Influence and Motivation

Many tools and technologies have been built to analyze the Android applications to make the world of users more and more secure. The question behind it is, "Is there any ideal tool and methodology which could provide the entire flow of the application?" In this line, framework like SOOT [6] provide analysis for the Java programs by implementing the framework for different types of applications like concurrent applications, point-to analysis. This framework has been used for static and dynamic analysis for intra and inter-procedural features. Flowdroid, [5] which is based on SOOT framework, is used for static taint analysis which may provide good precision and recall. But it is not easy to customize the framework like SOOT. As the applications are being increased exponentially, the possibility of penetration in the users' private data and resources also has increased. This possibility may occur substantially when an application is given a lot of permission than it requires. Apart from this another possibility is when one app requires the data from another app also increases malicious activities to be done through collusion. Another side is also there, and also when the coding standards are not good then also there is a possibility of information leakage [7] [8]. One of the technique to make the malicious code hidden from user or analyst is to use obfuscation. Even the anti malware systems are not able to detect the code if the code is highly obfuscated [9]. Many techniques have been implemented to make the anti malware systems more and more robust to detect malware like Androguard [10] [11].