1. Towards mining comprehensive android sandboxes

Le, T.-D.B. (1); Lingfeng Bao (2); Lo, D. (1); Debin Gao (1); Li Li (3)
Source: *2018 23rd International Conference on Engineering of Complex Computer Systems (ICECCS). Proceedings*, p 51-60, 2018; **ISBN-13:** 978-1-5386-9341-4; **DOI:** 10.1109/ICECCS2018.2018.00014; **Conference:** 2018 23rd International Conference on Engineering of Complex Computer Systems (ICECCS), 12-14 Dec. 2018, Melbourne, VIC, Australia; **Publisher:** IEEE Computer Society, Los Alamitos, CA, USA
Author affiliation: (1) Singapore Management University, School of Information Systems, Singapore (2) Zhejiang University City College, School of Computer and Computing Science, China (3) Monash University, Faculty of Information Technology, Clayton, VIC, Australia
Abstract: Android is the most widely used mobile operating system with billions of users and devices. The popularity of Android apps have enticed malware writers to target them. Recently, Jamrozik et al. proposed an approach, named Boxmate, to mine sandboxes to protect Android users from malicious behaviors. In a nutshell, Boxmate analyzes the execution of an app, and collects a list of sensitive APIs that are invoked by that app in a monitoring phase. Then, it constructs a sandbox that can restrict accesses to sensitive APIs not called by the app. In such a way, malicious behaviors that are not observed in the monitoring phase - occurring, for example, due to malicious code injection during an attack - can be prevented. Nevertheless, Boxmate only focuses on a specific API type (i.e., sensitive APIs); it also ignores parameter values of many API methods and requested permissions during the execution of a target app. As a result, Boxmate is not able to detect malicious behaviors in many cases. In this work, we address the limitation of Jamrozik et al.'s work by considering input parameters of many different types of API methods for mining a more comprehensive sandbox. Given a benign app, we first extract a list of Android permissions that the app may request during its execution. Next, we leverage an automated test case generation tool, named Droidbot, to generate a rich set of GUI test cases for exploring behaviors of the app. During the execution of these test cases, we analyze the execution of four different types of API methods. Furthermore, we record input parameters to these API methods, and classify those into four different categories. We leverage the collected parameter values, and the list of requested permissions to create a sandbox that can protect users from malicious behaviors. Our experiments on 25 pairs of real benign and malicious apps show that our approach is more effective than the coarse-and fine-grained variants of Boxmate by 267.37% and 81.64% in terms of F-measure respectively. (0 refs)
Inspec controlled terms: Android (operating system) - application program interfaces - graphical user interfaces - invasive software - mobile computing - program testing
Uncontrolled terms: comprehensive android sandboxes - widely used mobile operating system - Android apps - Android users - malicious behaviors - Boxmate analyzes - monitoring phase - malicious code injection - specific API type - requested permissions - target app - comprehensive sandbox - benign app - Android permissions - automated test case generation tool - GUI test cases - collected parameter values - malicious apps - sensitive API - Boxmate
Classification Code: C6130S Data security - C6150G Diagnostic, testing, debugging and evaluating systems - C6150J Operating systems - C6180G Graphical user interfaces - C6190V Mobile, ubiquitous and pervasive computing
IPC Code: G06F3/048 - G06F9/44 - G06F9/46 - G06F11/36 - G06F21/00
Treatment: Practical (PRA)
Database: Inspec
Data Provider: Engineering Village