Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 311 (2015) 3–14

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

AnaDroid: Malware Analysis of Android with User-supplied Predicates

Shuying Liang,[1,2] Matthew Might[1]

School of Computing, University of Utah
Salt Lake City, Utah, USA

David Van Horn[1]

College of Computer and Information Sciences
Northeastern University
Boston, Massachusetts, USA

Abstract

Today's mobile platforms provide only coarse-grained permissions to users with regard to how third-party applications use sensitive private data. Unfortunately, it is easy to disguise malware within the boundaries of legitimately-granted permissions. For instance, granting access to "contacts" and "internet" may be necessary for a text-messaging application to function, even though the user does not want contacts transmitted over the internet. To understand fine-grained application use of permissions, we need to statically analyze their behavior. Even then, malware detection faces three hurdles: (1) analyses may be prohibitively expensive, (2) automated analyses can only find behaviors that they are designed to find, and (3) the maliciousness of any given behavior is application-dependent and subject to human judgment. To remedy these issues, we propose semantic-based program analysis, with a human in the loop as an alternative approach to malware detection. In particular, our analysis allows analyst-crafted semantic predicates to search and filter analysis results. Human-oriented semantic-based program analysis can systematically, quickly and concisely characterize the behaviors of mobile applications. We describe a tool that provides analysts with a library of the semantic predicates and the ability to dynamically trade speed and precision. It also provides analysts the ability to statically inspect details of every suspicious state of (abstract) execution in order to make a ruling as to whether or not the behavior is truly malicious with respect to the intent of the application. In addition, permission and profiling reports are generated to aid analysts in identifying common malicious behaviors.

Keywords: static analysis, human analysis, malware detection

1 Introduction

Google's Android is the most popular mobile platform, with a share of 52.5% of all smartphones [10]. Due to Android's open application development community,

[1] Supported by the DARPA Automated Program Analysis for Cybersecurity Program.

[2] An extended report is available: http://matt.might.net/a/2013/05/25/anadroid/

http://dx.doi.org/10.1016/j.entcs.2015.02.002

1571-0661/© 2015 Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

more than 400,000 apps are available with 10 billion cumulative downloads by the end of 2011 [9].

While most of those third-party applications have legitimate reasons to access private data, the grantable permissions are too coarse: malware can hide in the cracks. For instance, an app that should only be able to read information from a specific site and have access to GPS information must necessarily be granted full read/write access to the entire internet, thereby allowing a possible location leak over the net. Or, a note-taking application can wipe out SD card files when a hidden trigger condition is met. Meanwhile, a task manager that requests every possible permission can be legitimately benign.

To understand fine-grained use of security-critical resources, we need to statically analyze the application with respect to what data is accessed, where the sensitive data flows, and what operations have been performed on the data (*i.e.*, determine whether the data is tampered with). Even then, automated malware detection faces three hurdles: (1) analyses may be prohibitively expensive, (2) automated analyses can only find behaviors that they are designed to find, and (3) the maliciousness of any given behavior is application-dependent and subject to human judgment.

In this work, we propose semantics-based program analysis with a human in the loop as an alternative approach to malware detection. Specifically, we derive an analytic engine, an abstract CESK* machine based on the design methodology of Abstracting Abstract Machines (AAM) [20] to analyze object-oriented bytecode. Then we extend the foundational analysis to analyze specific features: multiple entry points of Android apps and reflection APIs. Finally, we describe a tool that provides analysts with a library of semantic predicates that can be used to search and filter analysis results, and the ability to dynamically trade speed and precision. The tool also provides analysts the ability to statically inspect details of every suspicious state of (abstract) execution in order to make a ruling as to whether or not the behavior is truly malicious with respect to the intent of the application. Human-oriented, semantics-based program analysis can systematically characterize the behaviors of mobile applications.

Overview

The remainder of the paper is organized as follows: Section 2 presents the syntax of an object-oriented byte code, and illustrates a finite-state-space-based abstract interpretation of the byte code. Section 3 discusses analysis techniques to analyze Android-specific issues: multiple entry points and reflection APIs. Section 4 presents the tool implementation with user-supplied predicates. Section 5 discusses related work, and Section 6 concludes.

2 Semantic-based program analysis

Android apps are written in Java, and compiled into Dalvik virtual machine byte code (essentially a register-based version of Java byte code). In this section, we present how to derive an analysis for a core object-oriented (OO) byte code language

Download English Version:

https://daneshyari.com/en/article/422770

Download Persian Version:

https://daneshyari.com/article/422770

Daneshyari.com