Università degli Studi di Padova

Department of Mathematics

Master Degree in Computer Science

All You Ever Wanted to Know About Dynamic Taint Analysis and Forward Symbolic Execution

(but might have been afraid to ask)

Matteo Di Pirro
1154231

Academic Year 2016-2017

Contents

1 Introduction

1.1 Static and Dynamic Analysis

Dynamic analysis is the analysis of the properties of a running program. In contrast to static analysis, which examines a program's text to derive properties that hold for all executions, dynamic analysis derives properties that hold for one or more executions by examination of the running program. While dynamic analysis cannot prove that a program satisfes a particular property, it can detect violations of properties as well as provide useful information to programmers about the behavior of their programs. Although dynamic analysis provides a powerful mechanism for relating program inputs, this dipendence from the inputs makes it incomplete. Viewed in this light, dynamic and static analysis might be better termed "input-centric" and "program-centric" analysis respectively. [1]

Dynamic analysis is attractive because it allows us to reason about actual executions, and thus can perform precise security analysis based upon run-time information. Further, dynamic analysis is simple: we need only consider facts about a single execution at a time.

1.2 Questions about user input

The two analyses can be used in conjunction to build formulas representing only the parts of an execution that depend upon tainted values.

1.2.1 Is the final value affected by user input? ⇒ Dynamic Taint Analysis

Dynamic taint analysis runs a program and observes which computations are affected by predefined taint sources such as user input. . Any program value whose computation depends on data derived from a taint source is considered *tainted*. Any other value is considered untainted.

1.2.2 What input will make execution reach this line of code? ⇒ Forward Symbolic Execution

Dynamic forward symbolic execution automatically builds a logical formula describing a program execution path, which reduces the problem of reasoning about the execution to the domain of logic.

Bibliography

[1] Thomas Ball. "The concept of dynamic analysis". In: *Software Engineering—ESEC/FSE'99*. Springer. 1999, p. 216 (cit. on p. 2).