Assisting Fuzzing with Symbolic Execution

Søren Lund Jensen

4. april 2017

Indhold

1 Introduction

An ever-present danger in today's society is memory corruption vulnerabilities in software. An attacker could, did he know of these vulnerabilities, exploit them in order to access confidential informations, and as computer processing, and connecting continues to be on the rise, playing a major role in present day, patching these vulnerabilities has to be a priority. This, of course, cannot be done without first discovering the bugs. A variety of tools exists, with the purpose of doing so, but as the bugs are often very specific, and/or wide-spread, creating a silver bullet is hard, if not impossible.

2 Concept

2.1 Fuzzing

Stemming from the early years of punch-card-programming, a technique, known as fuzzing exists. This technique works by feeding random input to a program, at a very high rate, some of which will hit specific vulnerabilities in said program. Upon vulnerability-hit, a Fuzzer logs the vulnerability, along with information about where the vulnerability occurred, and which input triggered it.

An advantage, as well as a drawback of most fuzzers is their execution method. They are as little invasive as possible, as to prioritize speed. This means that a typical fuzzer does not analyse a fuzzed application - instead directly executing the application with random input, which is immensely faster than finding qualified input variables, based on an application analysis.

Features of Fuzzing

Genetic Fuzzing
sada
sadsa