Impact factors for severity assessment of bugs discovered via compositional symbolic execution
Master Thesis

Starting: Immediately

Context

Symbolic execution [1] is a powerful way to analyze programs by executing them using “symbolic” values instead of concrete values, and exploring as many program paths as possible within a given amount of time. It is also an effective way to generate inputs (exploits) that lead to potential vulnerabilities in a program [2]. However, symbolic execution suffers from path-explosion problem as well as the bottlenecks associated with underlying SMT solvers. In order to solve these problems, a compositional approach to symbolic execution has been proposed in the past [3]. MACHE [4] is such a compositional symbolic execution framework that applies “divide-and-conquer” approach to a program with smaller components that interact based on input pre-conditions and output post-conditions. Despite the decrease in size of queries, compositional symbolic execution might still be unable to achieve high path-coverage (eg. input-dependent loops).

When certain “discovered” vulnerabilities are not confirmed via higher compositional levels, we propose to assign them severity values based on various heuristics. These heuristics may be based on the structural properties of the program (eg. Functional composition, design of APIs, input grammar), peculiarities of development environments, trustworthiness of re-used components, or a combination of these factors. A complete compositional framework should, ideally, take into account all possible factors to assign severity to vulnerabilities, even if exploits cannot be automatically generated.

Goal

The goals of this thesis are as follows –

1. Identify research gaps in vulnerability assessment in symbolic execution, by means of a literature survey.
2. Identify impact factors for severity assessment, specific to the context of compositional symbolic execution.
3. Evaluate the effectiveness of severity assignment for compositional analysis, by comparing bug detection/fixing, both, with and without the complete compositional framework.
4. Present the claims, design and evaluation in a coherent manner, comparing to existing techniques in literature.

Workplan

In order to fulfill the above goals, your tasks will include the following –

1. Study symbolic execution and compositional analysis, including their implementations and target-specific techniques.
2. Familiarize yourself with MACHE, our in-house tool for compositional symbolic execution of C programs.
3. Study existing literature on severity-scoring systems, such as CVSS [5] and AWARE [6], as inspirations for vulnerability assessment in incomplete symbolic execution.
4. Identify structural aspects of SUTs (*impact factors*) that may be suitable for assigning severity to vulnerabilities discovered with MACKE.

5. State hypotheses using impact factors to calculate severity of discovered vulnerabilities, using a *severity function*.

6. Test various hypotheses by application on our existing database of (labeled) vulnerability reports in open-source programs –
   i. Prepare and analyze the programs with MACKE.
   ii. Present the analysis results (with discovered vulnerabilities) to students (with experience in software security concepts), along with hints about some impact factors such as list of interfaces, call-graph depth, length of input buffers etc.
   iii. Observe and record bug prioritization by the participants.
   iv. Combine the feedback from participants with alternative severity ranking methods, including manual entries from Bugzilla [7].

7. Update the hypotheses and measure effectiveness of automated severity assignment process on real-world examples.

8. Completely integrate severity assessment tool into the MACKE framework.

**Eligibility**

An ideal candidate should –

1. Have prior programming experience in Python.
2. Preferably have prior experience in C/C++.
3. Have at least beginner to intermediate level skills in HTML5 and Javascript.

We are interested in highly motivated students for this study, who will help further our research in the field of automatic analysis of programs for security vulnerabilities. We expect the student to produce high-quality deliverables, both in terms of scientific and engineering contributions.

**References**