Master Thesis Topic
Behavioral Clustering-Guided Fuzz Testing

Motivation and Background
Fuzzing or fuzz testing [1] is an established technique that aims to discover unexpected program behavior (e.g., bugs, security vulnerabilities, or crashes) by feeding automatically generated data into a system under test (SUT). Current state-of-the-art techniques typically employ a genetic algorithm and produce new test cases by mutating previously executed test inputs using generic mutation operators (e.g., bit-flips or block deletions). However, one of the drawbacks of these generic, uninformed mutation-based techniques is that many test cases end up exercising the same (or similar) program behavior. In the worst case, the majority of test cases ends up executing error-handling code in the input parser, while only a small number of inputs actually exercise the main functionality of the SUT. A possible idea to alleviate this problem is to cluster test cases based on their execution behavior and learn the structural properties of the inputs that determine cluster membership (e.g., using invariants [2] or constraint-learning [3]), which may be exploited to guide subsequent test case generation. For example, the technique may discover two clusters (Cluster 1: “The error-handling code of the parser is executed”, Cluster 2: “The main functionality is executed”), where inputs in Cluster 1 are all negative and inputs in Cluster 2 are non-negative. Thus, in order to generate test cases that target the main functionality, only non-negative inputs should be produced.

Goals
The goal of this thesis is to explore clustering-based fuzzing techniques and compare the developed approach against current state-of-the-art methods.

Description of the Task
The specific tasks are:
- Getting familiar with the current state-of-the-art in fuzz testing
- Develop a technique to guide test input generation based on clustering methods
- Perform an experimental evaluation of the implemented approach and compare against existing fuzzing tools on a suitable benchmark

Research Type
Theoretical Aspects:  *****
Industrial Relevance:  *****
Implementation:  *****

Prerequisite
The student should be enrolled in the master of computer science program, and has completed the required course modules to start a master thesis.

Skills required
Programming skills (e.g., in Java, Python, or C/C+), understanding of, or willingness to learn, the theoretical foundations (e.g., clustering, invariant learning, constraint learning) and software engineering methods (e.g., fuzz testing) needed for the project.

Contact
Hoang Lam Nguyen (nguyehoa@informatik.hu-berlin.de)
Software Engineering Group, Institut für Informatik, Humboldt-Universität zu Berlin

References