Software Model Checking Improving Security of a Billion Computers

Patrice Godefroid

Microsoft Research
pg@microsoft.com

Abstract. I will present a form of software model checking that has improved the security of a billion computers (and has saved Microsoft millions of dollars). This form of software model checking is dubbed whitebox fuzz testing, and builds upon recent advances in systematic dynamic test generation (also known as DART) and constraint solving. Starting with a well-formed input, whitebox fuzzing symbolically executes the sequential program under test dynamically, and gathers constraints on inputs from conditional statements encountered along the execution. The collected constraints are negated systematically one-by-one and solved with a constraint solver, yielding new inputs that exercise different execution paths in the program. This process is repeated using novel state-space exploration techniques that attempt to sweep through all (in practice, many) feasible execution paths of the program while checking simultaneously many properties. This approach thus combines program analysis, testing, model checking and automated theorem proving (constraint solving).

Whitebox fuzzing has been implemented in the tool SAGE, which is optimized for long symbolic executions at the x86 binary level. Over the past 18 months, SAGE has been running on hundreds of machines and has discovered many new expensive security-critical bugs in large shipped Windows applications, including image processors, media players and file decoders, that are deployed on more than a billion computers worldwide. SAGE is so effective in finding bugs missed by other techniques like static analysis or blackbox random fuzzing that it is now used daily in various Microsoft groups.

This is joint work with Michael Levin (Microsoft CSE) and other contributors.