500 Machine-Years of Software Model Checking and SMT Solving

Patrice Godefroid

Microsoft Research, pg@microsoft.com

Abstract. I will report on our experience running SAGE for over 500-machine years in Microsoft's security testing labs. SAGE is a whitebox fuzzing tool for security testing. It performs symbolic execution dynamically at the binary (x86) level, generates constraints on program inputs, and solves those constraints with an SMT solver in order to generate new inputs to exercise new program paths or trigger security vulnerabilities (like buffer overflows). 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).

Since 2009, SAGE has been running 24/7 on average 100+ machines automatically "fuzzing" hundreds of applications. This is the largest computational usage ever for any SMT solver, with over 4 billion constraints processed to date. In the process, SAGE found many new security vulnerabilities (missed by blackbox fuzzing and static program analysis) and was credited to have found roughly one third of all the bugs discovered by file fuzzing during the development of Microsoft's Windows 7, saving millions of dollars by avoiding expensive security patches to nearly a billion PCs.

In this talk, I will present the SAGE project, highlight connections with program verification, and discuss open research challenges.

This is joint work with Michael Levin, David Molnar, Ella Bounimova, and other contributors.