Formal Analysis of x86 Machine-Code Programs

- Shilpi Goel and Warren A. Hunt, Jr.

[Source Code] https://github.com/acl2/acl2/tree/master/books/projects/x86isa

[Documentation] www.cs.utexas.edu/users/moore/acl2/manuals/current/manual/?topic=ACL2____X86ISA

Objective

Develop general-purpose tools and techniques to increase software reliability

State-of-the-Art

- Bug-hunting tools are limited in their scope
- Formal verification tools have a high user overhead; as such, formal software models are often simplified

Our Proposal

- Develop a *practical, general-purpose* formal verification framework to verify complex program properties *without any loss of accuracy or expressiveness*
- Target: **x86 Machine-Code Programs**
- Formal Tool Used: ACL2 Theorem Prover
  - Industrial-strength tool
  - Routinely used for hardware and software verification

Why Not Verify High-Level Programs?

- Sometimes, high-level code is unavailable (e.g., malware, executables on the Internet)
- High-level verification frameworks do not address compiler bugs
- Would need to build verification frameworks for many high-level languages, whereas machine-code verification is applicable whenever a program compiles down to the supported hardware platform

Approach

(I) Develop a formal model of the x86 ISA
(II) Develop techniques for program analysis
(III) Employ (I) & (II) to verify real x86 programs

(I) x86 ISA Model

- A formal model of the x86 ISA provides semantics to x86 machine -code programs.
- *Current status:* 400+ opcodes, includes all addressing modes, and system features like paging and segmentation
- Validated regularly via co-simulations against a physical x86 processor

(II) Techniques for Program Analysis

- Reason about straight-line x86 machine-code *completely* automatically; useful for compositional verification
- General-purpose libraries to reason about supervisor-mode programs
  - E.g.: tactics to reason about programs that modify low-level ISA data structures to perform security-critical tasks, such as altering permissions

Future Work

- Extend x86 ISA model to support caches, interrupts, and multiprocessor programs
- Further automate reasoning about x86 machine-code programs
- Collaborate with the OS community (e.g., FreeBSD) to *identify* security-critical code, *specify* its behavior, and then *verify* it against this specification

(III) Case Studies

- *User-mode Program Verification:*
  - Completely automatic verification of a program written using obfuscated code that employs complicated bit-vector operations to compute the *population-count* of the input
  - Largely automatic verification of a *word-count* program that computes the number of characters, words, and lines in the input file
- *Supervisor-mode Program Verification:*
  - Largely automatic verification of a *zero-copy* program that implements data copying via the Copy-on-Write technique; it modifies the virtual memory abstraction to give the illusion that data was indeed copied from the source to the destination

Interested in meeting the PIs? Attach post-it note below!

National Science Foundation
WHERE DISCOVERIES BEGIN

NSF Secure and Trustworthy Cyberspace Inaugural Principal Investigator Meeting
Nov. 27 -29th 2012
National Harbor, MD