Verifying enclave systems with Serval

Luke Nelson

w/ James Bornholt, Ronghui Gu, Andrew Baumann, Emina Torlak, Xi Wang

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY of WASHINGTON

UNSAT

Enclave systems are hard to get right

- Correctness of low-level systems code is critical for security
- Many different kinds of bugs are security vulnerabilities:
  - **Low-level bugs**: e.g., buffer overflow or use-after-free
  - **Logic bugs**: implementation does something unintended
  - **Design bugs**: intended design of the system is not secure
- Each can be exploited to compromise the entire system

Eliminating bugs with formal verification

- **Goal**: prove absence of low-level, logic, and design bugs
  - Functional correctness (state-machine refinement)
  - Noninterference
- **Approach**: Use automated verification techniques
  - Low proof burden: symbolic evaluation / SMT solvers
  - Bounded loops in code
  - Limitations: no concurrency or side channels

Challenges

- Difficulty of building verifiers
  - Need detailed RISC-V machine model
  - Need to reason at ISA level
- Difficulty of scaling to practical systems
  - Symbolic evaluation
  - SMT solving

Serval: A framework for verifying low-level systems

- Built on top of Rosette
- Lift ISA interpreter into verifier
  - Easier to write and test
  - Supports LLVM IR and RISC-V
- Use symbolic profiling to identify verification bottlenecks
- Use symbolic optimizations to scale verification

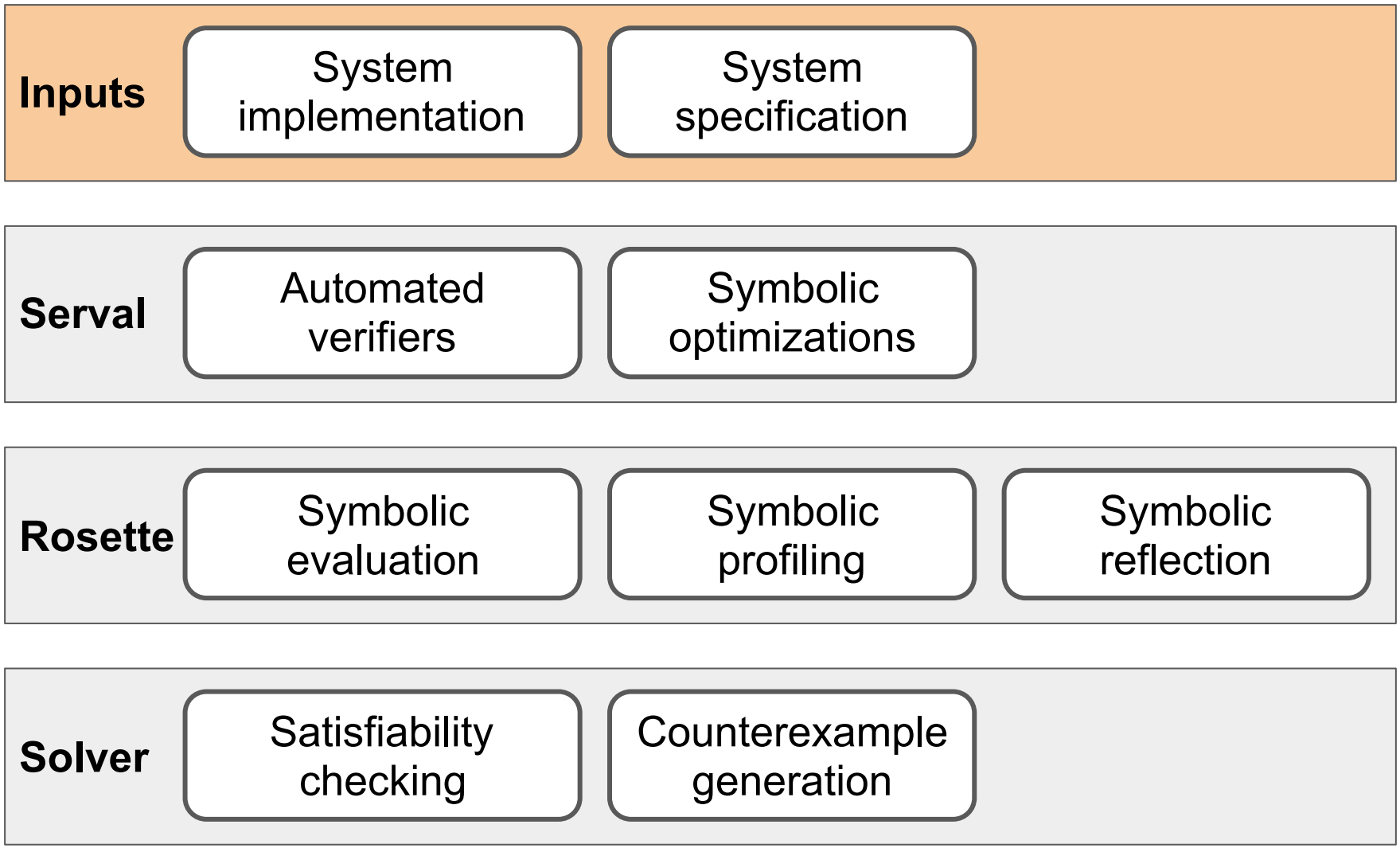
Main Results

- Used to find and fix low-level and design bugs in Keystone
- Applied to CertiKOS and Komodo, previously manually verified using Coq and Dafny
- Found and fixed 15 Linux BPF JIT bugs, all now upstreamed.
  - https://git.kernel.org/linus/1e692f09e091
  - https://git.kernel.org/linus/46dd3d7d287b
  - https://git.kernel.org/linus/68a8357ec15b
  - https://git.kernel.org/linus/6fa632e719ee

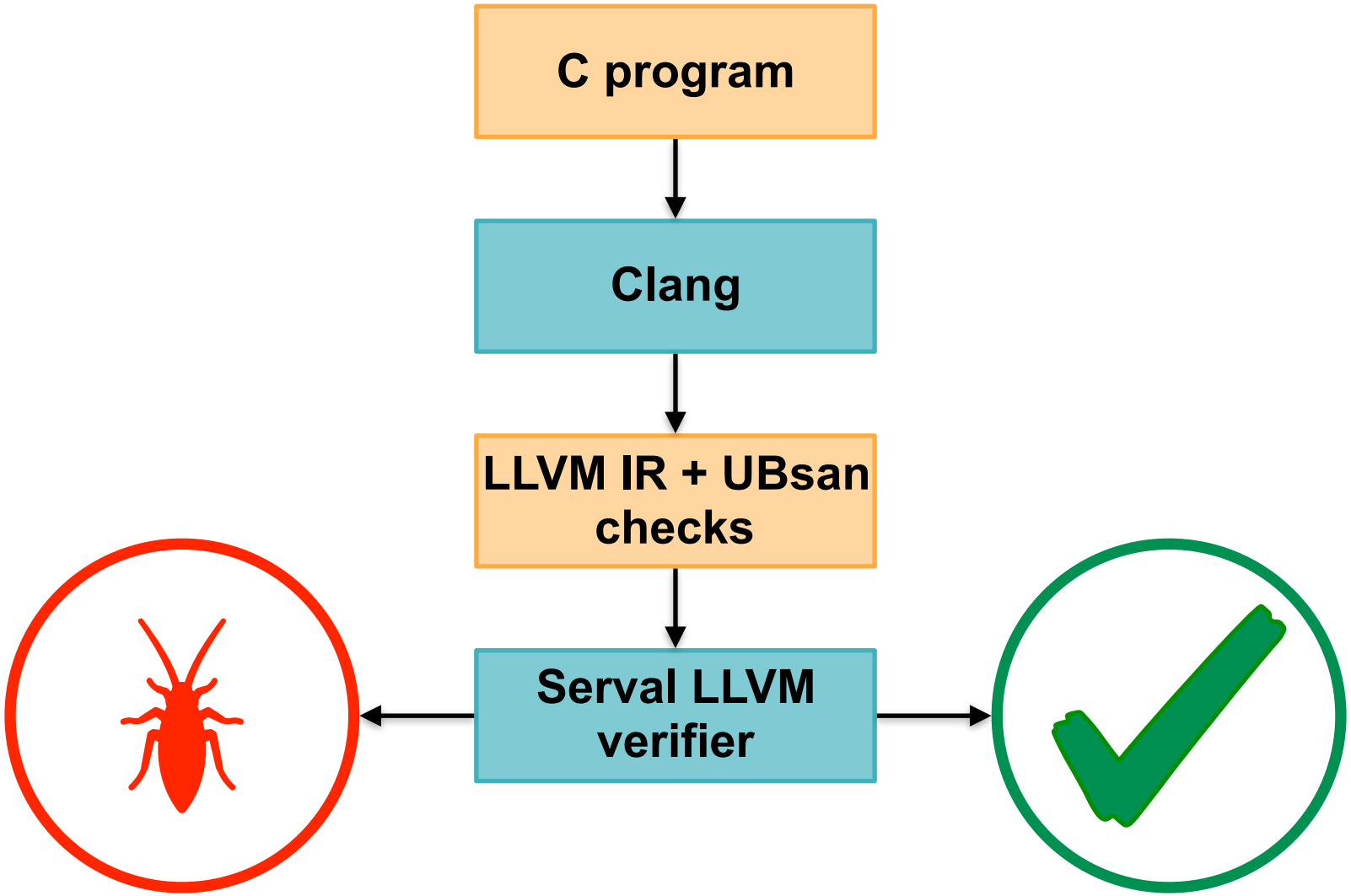
Outline

- Design
- Workflow
- Demo

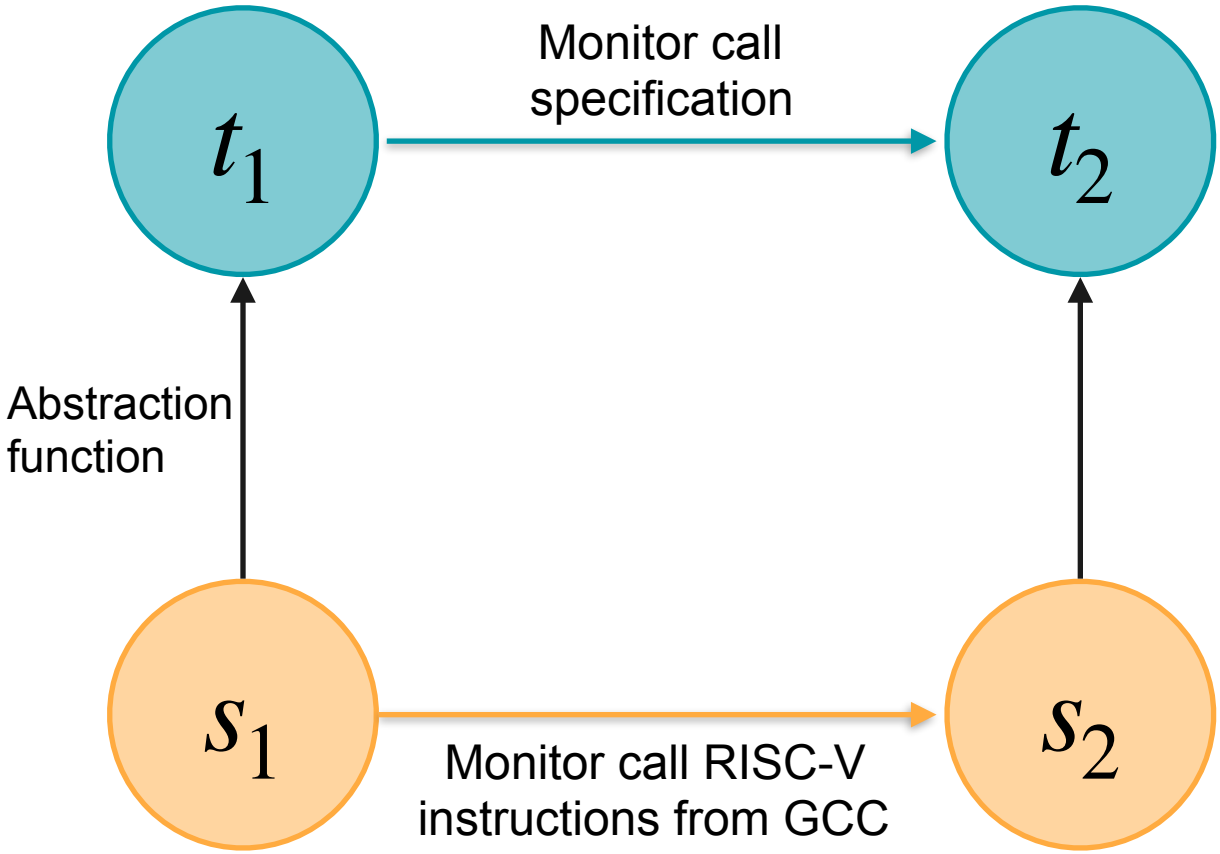
Design


Inputs
System implementation
System specification
Serval
Automated verifiers
Symbolic optimizations
Rosette
Symbolic evaluation
Symbolic profiling
Symbolic reflection
Solver
Satisfiability checking
Counterexample generation

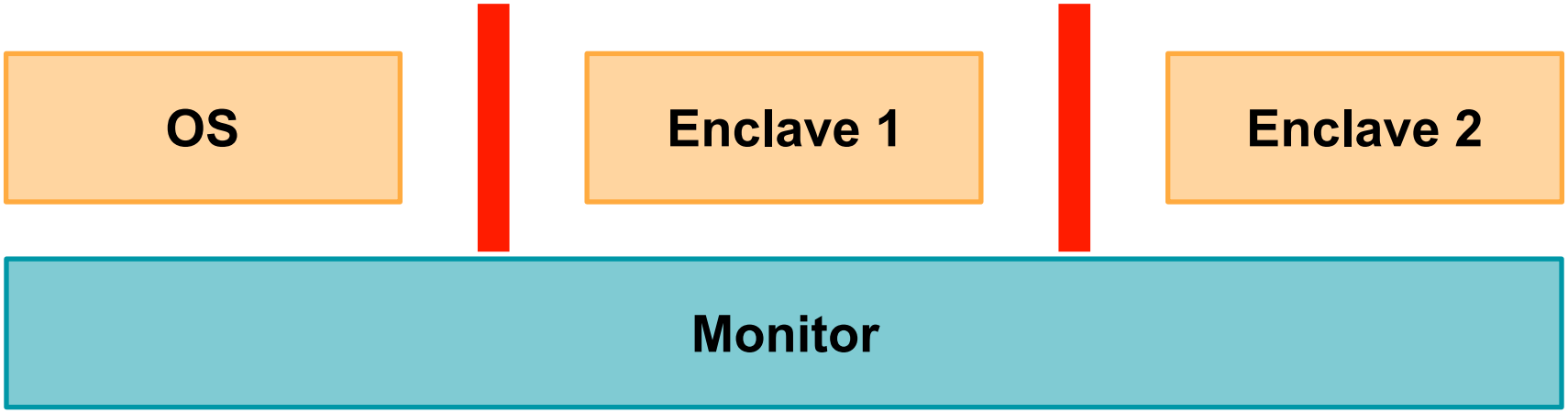
Proving absence of low-level bugs



Proving functional correctness



Proving noninterference



- Example: bogus monitor call that returns enclave secrets
- Integrity — OS should not modify be able to modify enclave-visible state
- Confidentiality — Behavior of OS is independent of enclave secrets

Example: Komodo

- Komodo is a verified software enclave monitor for ARM
- We have ported to RISC-V and verified using Serval

- Two demonstrations:
  - Buffer overflow
  - Functional correctness violation

Demo

Conclusion

- Automated verification is effective at eliminating bugs in low-level systems
- If you are building enclave systems, talk to us!
- Paper to appear at SOSP'19
- Code will be released shortly

`https://serval.unsat.systems/`

UNSAT