Toward Efficient Formal Verification of Reference Monitors for Isolated Execution

Ryo Nakashima
The University of Tokyo
nakashima@os.ecc.u-tokyo.ac.jp

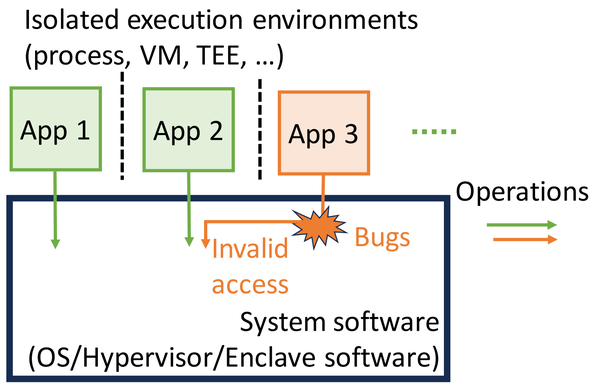
Takahiro Shinagawa
The University of Tokyo
shina@is.s.u-tokyo.ac.jp

1 Background

- Cloud computing has been widely used
- Isolation between apps is becoming more important

System software that provides isolation is becoming larger and more complex

- **High risk of potential bugs**
- **hard to verify the correctness**



2 Previous Work

- Verification targets
  - Operating System Kernels
  - Hypervisors
  - Enclave managers

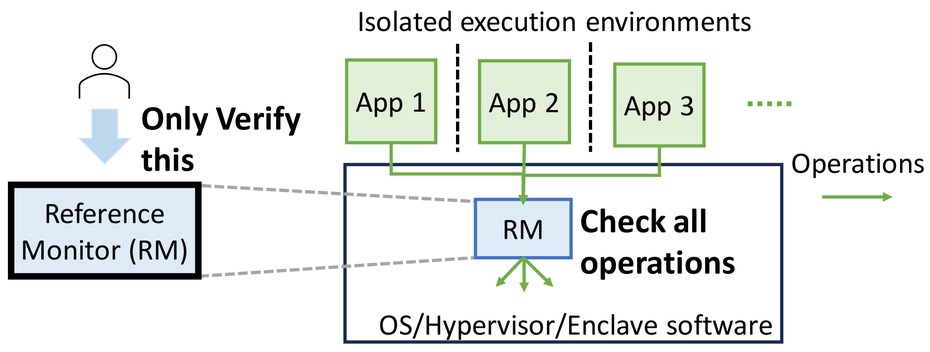
Problem: Often targets huge software as a whole

- Verification properties
  - The correctness of entire system software
  - Assume no external attacks

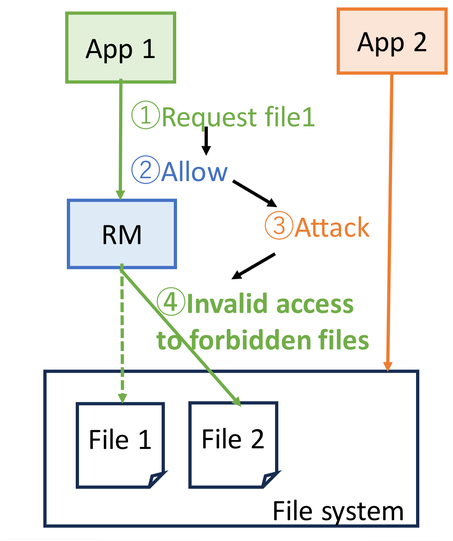
Problem: Often targets only general properties

3 Proposal: Formal Verification of Reference Monitors

- **Approach: Focus on security-critical components but support diverse attack patterns**



(1) Verify properties to prevent vulnerabilities common to RMs
=> **time-of-check to time-of-use (TOCTTOU)** is our 1st target

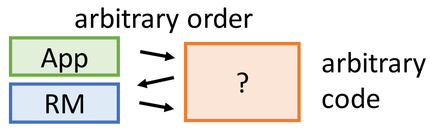


Policy

	App 1
File 1	○
File 2	×

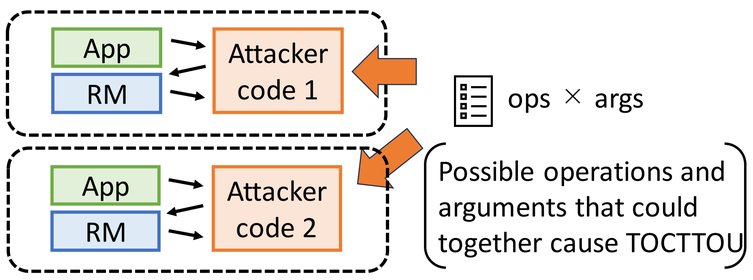
App1 operation
App2 operation
Execution order

Problem: attackers run arbitrary code



Proposal: two-step exhaustive verification

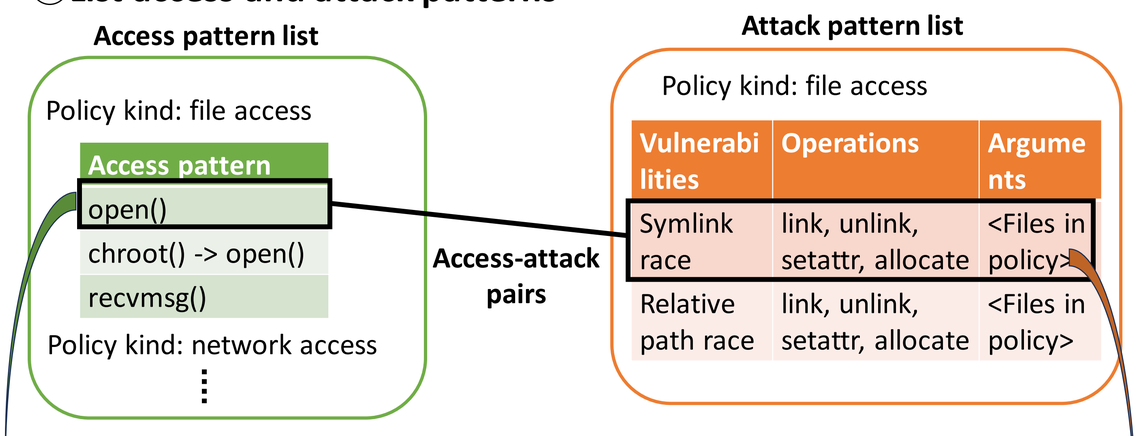
Step 1: Generate attack code (all combinations of ops and args)



Step 2: test all execution orders (for each pair)

(2) Generalize to diverse attack patterns (and implicit accesses)
=> Verification framework for reference monitors

①List access and attack patterns

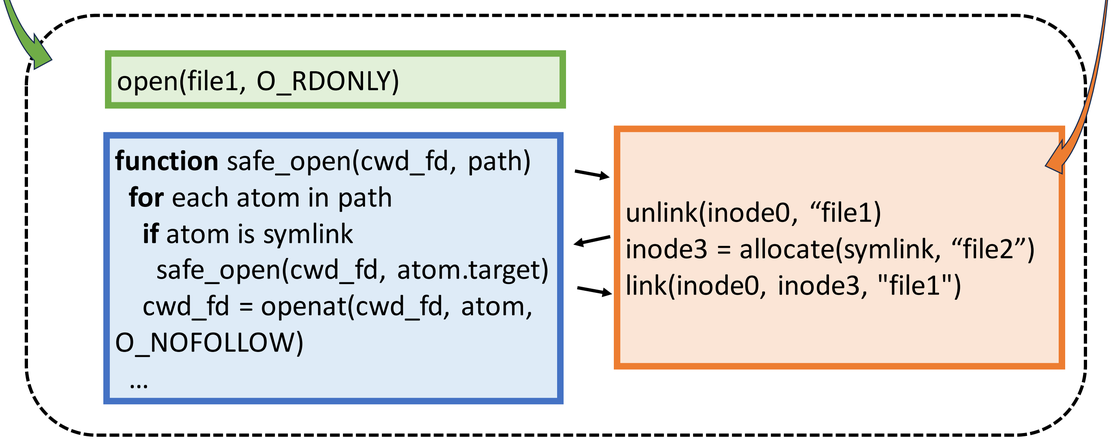


②Generate all access-attack pairs given a policy

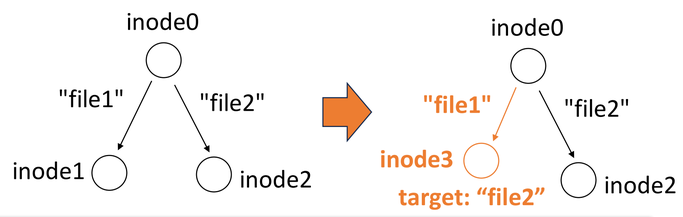
Policy (file access)

	App 1
File 1	○
File 2	×

All combinations of ops and args (“Step 1” on the left side)



③Verify all the access-attack pairs



4 Evaluation

Case study:
- File access policy
- 1 access pattern (direct open)
- 1 vulnerability (symlink race)
- 1 attack pattern (unlink, allocate, link)

Symlink-race-free RM → **Verified**

Max preemption	# of interleavings	Verification time
2	224	21.58
3	1444	179.2

Symlink-race-vulnerable RM → **Race detected**

5 Implementation

- Implementation in Rust, to leverage its memory safety
- Loom model checker, to permute concurrent executions
- Mock file system, to use data structure for permutation hooks

6 Future Work

- Encompass more kinds of vulnerabilities in our framework
- Evaluate the whole framework in terms of security and verification costs