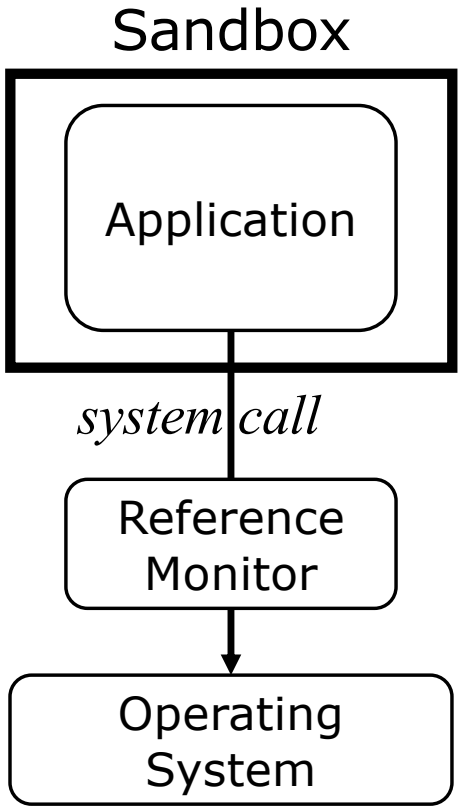
vRM: Verifying Reference Monitors via Exhaustive Access Pattern Generation

Ryo Nakashima, Takahiro Shinagawa*
The University of Tokyo

Sandbox

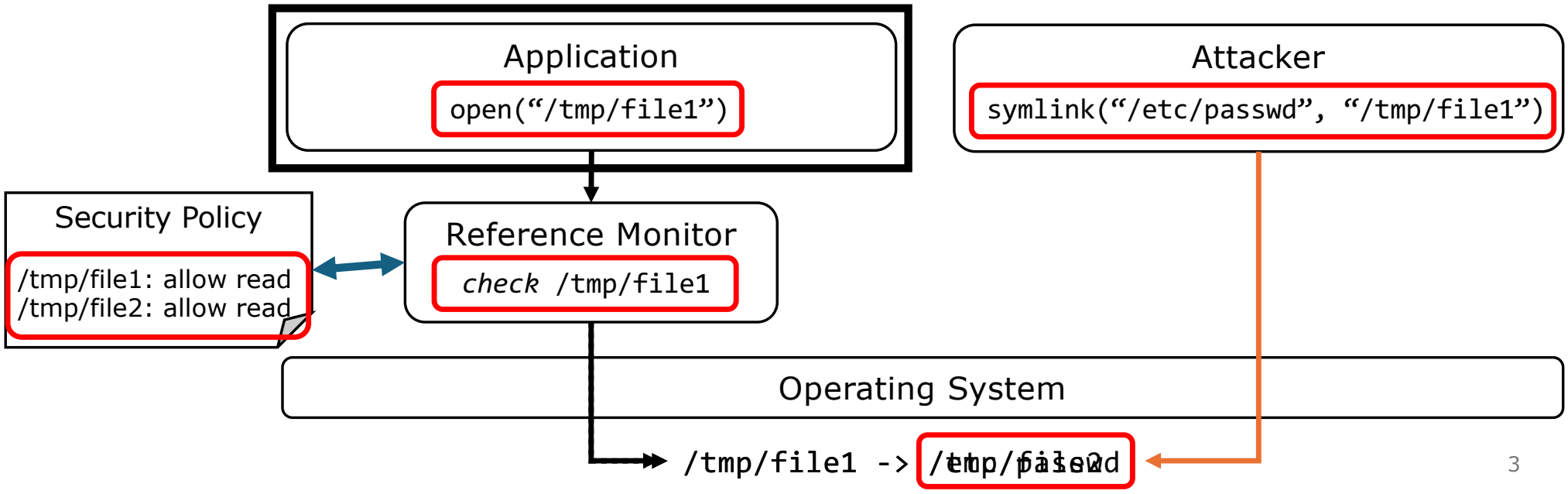
- An isolated execution environment
  - Designed to minimize the impact of attacks
    - *e.g.* Google Chrome, gVisor
- User-space system-call-level sandboxing
  - Flexible, efficient, and extensible
    - *e.g.* ptrace-based [1], Intel PKU-based [2]
- **Reference monitor** is a key component
  - Interposes on system calls
  - Enforces predefined security policies



[1] T. Kim and N. Zeldovich, "Practical and Effective Sandboxing for Non-root Users," USENIX ATC '13.
[2] D. Schrammel et al. "Jenny: Securing Syscalls for PKU-based Memory Isolation Systems," USENIX Security Symposium 2022.

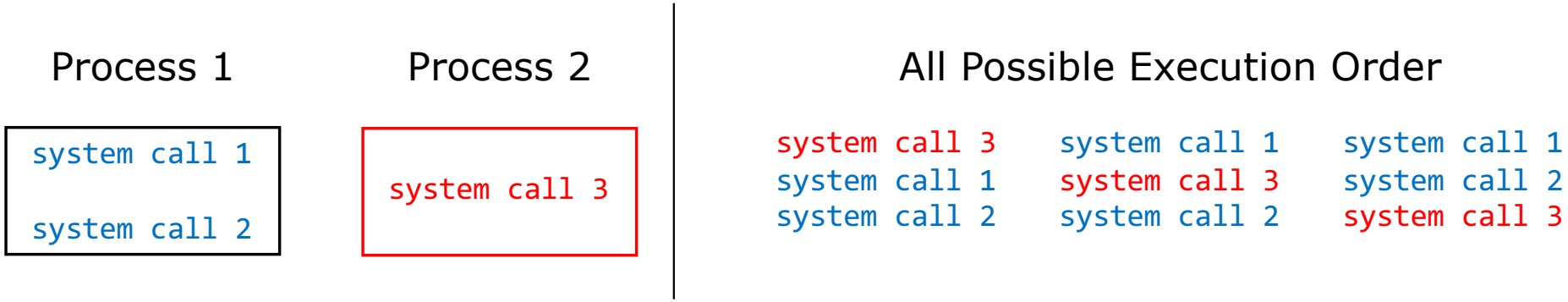
Time-of-Check to Time-to-Use

- Exploits the time window between the check and access
  - *After* the reference monitor' check
  - *Before* the actual access by the operating system



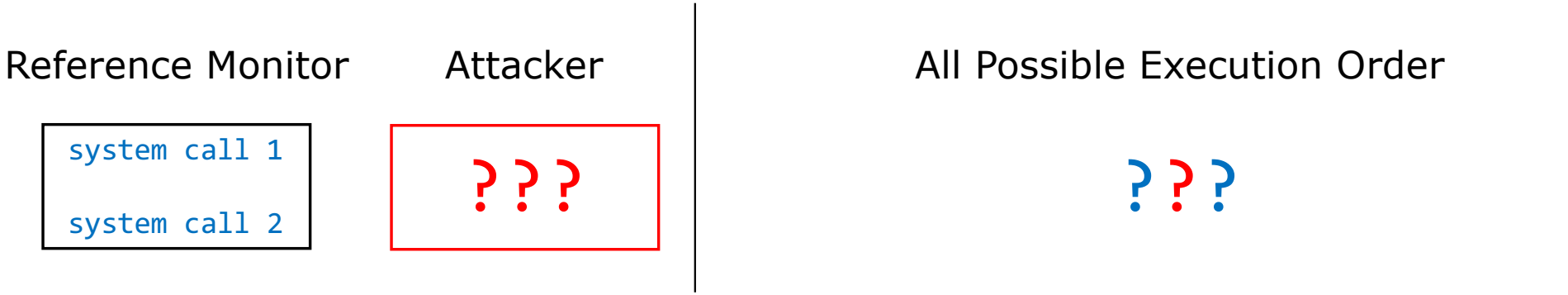
Model Checking

- A type of formal verification
  - Systematically explores all possible system states
- Effective for verifying concurrent processes
  - Detects subtle race conditions and interleaving errors



Verifying Reference Monitors

- Attackers can issue any system calls
  - In any order
- How can we model the attacker system call sequence?
  - It could originate from arbitrary code

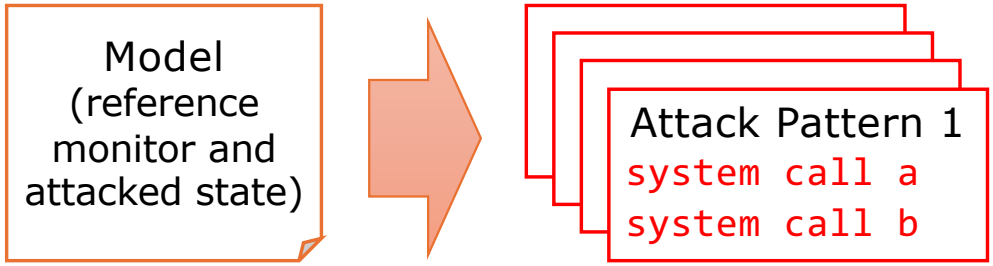


Related Work

- TOCTTOU countermeasures
  - Kernel-space reference monitors (*e.g.*, seccomp, SELinux)
    - Prevent TOCTTOU but are inflexible
  - User-space reference monitors (*e.g.*, Jenny, MBOX)
    - Flexible but require careful implementation
- Formal verification of reference monitors
  - Reference monitors for TEEs (*e.g.*, Komodo, VeriSMo)
    - Does not address TOCTTOU vulnerabilities

vRM: A two-step formal verification

1. Generates attack patterns exhaustively
   - Defines the reference monitor and attacked state
     - Using system call sequences and a file system model
   - Derives attack patterns with an SMT solver
2. Explores concurrent execution orderings
   - Applies model checking to analyze all possible interleavings



1. Generation Phase

All Possible Execution Order

```
system call a   system call a
system call b   system call 1
system call 1   system call b
system call 2   system call 2
```

.....

2. Verification Phase

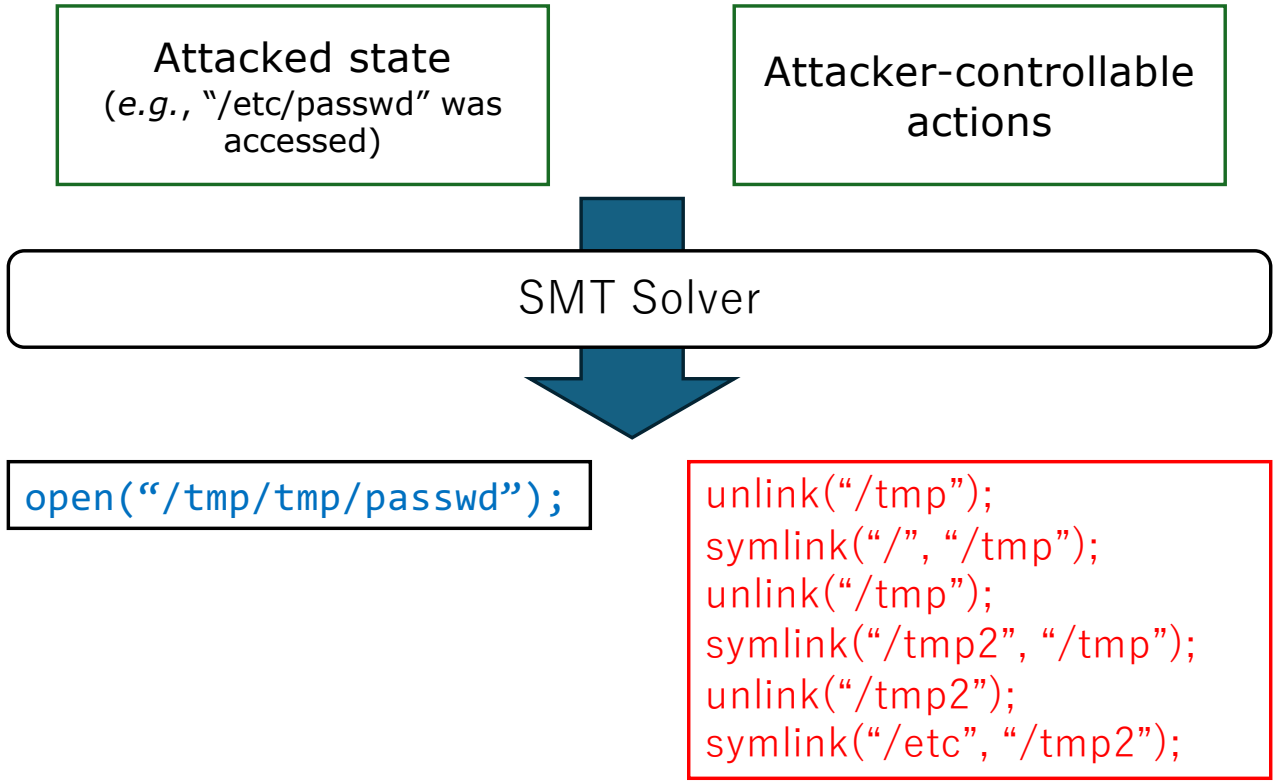
Generation Phase

Directory Structure

```
/
+--- tmp
|     +--- file1
+--- tmp2
+--- etc
      +--- passwd
```

Security Policy

```
/tmp/file1: allow
/etc/passwd: deny
```



Verification Phase

Explore All Possible Execution Order

```
open(“/tmp/passwd”);
symlink(“/etc”, “/tmp”);
```

```
symlink(“/etc”, “/tmp”);
open(“/tmp/passwd”);
```

```
open(“/tmp/passwd”);
symlink(“/”, “/tmp”);
symlink(“/etc/” /”/tmp”);
```

```
symlink(“/”, “/tmp”);
open(“/tmp/passwd”);
symlink(“/etc/” /”/tmp”);
```

```
symlink(“/”, “/tmp”);
symlink(“/etc/” /”/tmp”);
open(“/tmp/passwd”);
```

....

Detected simlink race

```
check(“/tmp/passwd”);
symlink(“/etc/passwd”, “/tmp/passwd”);
open(“/tmp/passwd”);
```

Implementation

- Attack pattern generator
  - Z3 SMT solver
    - Converts Z3 solutions into C system call sequences
- Verification:
  - CDSChecker model checker
    - Enables verification of C programs
  - mock file system
    - Provides the same interface as system calls

Evaluation of Verification

- Implemented a TOCTTOU-resistant algorithm [3]
  - performs all path component checks atomically
    - To eliminate TOCTTOU windows

- Formally verified its correctness
  - Total number of concurrent access timings: 2,350,240
  - Time required for verification: 3,509 seconds
    - Intel Core i5-12400F (6 cores, 2.5GHz), 8GB memory

[3] Tal Garfinkel. Traps and pitfalls: Practical problems in system call interposition based security tools. In Proc. NDSS 2023.

Conclusion

- vRM provides formal verification of reference monitors
  - Systematically generates attack patterns
  - Explores concurrency to detect vulnerabilities

- We implemented vRM
  - Uses the Z3 SMT solver for attack pattern generation
  - Uses the CDSChecker model checker to verity reference monitors
  - Employs a mock file system to simulate file system behavior

- Enhances security in sandboxed environments
  - Verified reference monitors with **no TOCTTOU vulnerabilities**