Operating System Transactions

Donald E. Porter, Indrajit Roy, and Emmett Witchel
The University of Texas at Austin
tx@cs.utexas.edu

Problem Statement

Secure sandboxing is hard to achieve

- Sandbox restricts access to system resources by untrusted app.
  - Enforces security policy on apps that can't be modified
- Lack of OS mechanisms to isolate system resource usage
- Concurrent accesses can be exploited by attackers
- No easy way to rollback system state in case of a breach
- Examples of sandboxing systems: Janus, Ostia, Systrace, Plash
  - Janus vulnerable to symbolic link races, argument races, etc.
  - Others act as proxy to OS, copying syscall parameters

Example

Time-of-check-to-time-of-use (TOCTTOU) Attacks

- Attacker exploits race condition to trick a setuid program
- Changes a symbolic link between check and use

Victim	Attacker
`if(access('foo')) {`	
	`symlink('secret','foo');`
`  fd=open('foo');`	
`  ...`	
`}`	

- No deterministic solution without changing API
- Current solutions are expensive, and yet probabilistic
- 400+ hits in National Vulnerability DB for "*symlink attack*"

State of the art

Proliferation of *ad hoc* solutions to races

- Linux has made numerous additions to file system API
  - openat, renameat, faccessat and ten more
- Signal handling API has been redesigned
  - sigaction, pselect, sigprocmask, epoll_wait, etc.
- Complex semantics, difficult to learn. No end in sight …

Approach

System transactions synchronize access to system resources

- Atomic and isolated access to files, memory allocation, etc.
- Users wrap system calls inside a transaction
- Simple API. Only three new system calls:
  - **sys_xbegin, sys_xend, sys_xabort**
- TOCTTOU can be solved deterministically with transactions

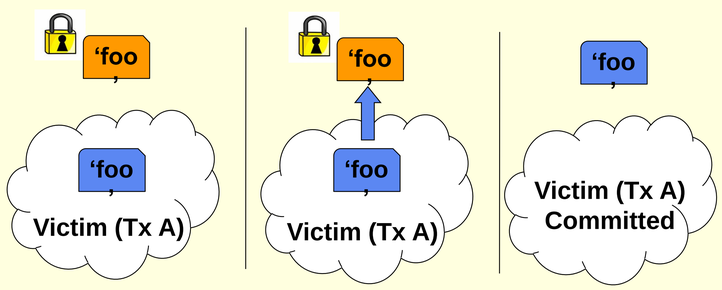
Victim	Attacker
	`symlink('secret','foo');`
`sys_xbegin();`	
`if(access('foo')) {`	
`  fd=open('foo');`	
`  ...`	
`}`	
`sys_xend();`	
	`symlink('secret','foo');`

- Similar in spirit to Quicksilver, TABS, Locus
  - Different semantics and implementation

Design

Version kernel data, detect conflicts and commit

- Transactions work on private copies of objects
- *Lazy version management* and *eager conflict detection*
- Locks held only to make copies and while committing
- Conflicts from non transactional threads are also detected
- At commit time atomically copy modifications to stable objects

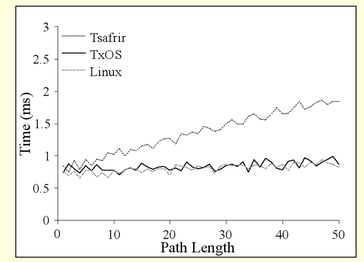


Results

- We have implemented system transactions in Linux 2.6.22.6
- Affectionately called '**TxOS**'

Solving TOCTTOU

- Tsafrir: user-level path resolution, probabilistic guarantees
  - Performance decreases with length of file name
- TxOS uses transactions, deterministic guarantees
  - Performance indistinguishable from Linux
  - Performance independent of file name length



Integration with Software Transactional Memory

- **Open Problem:** supporting system calls in TM
  - Sys. calls violate isolation: results visible to other threads
  - **Solved using system transactions**
- Experiment compares performance to locks in a web server
- Transactions provide better scalability than locks.