TxBox: Building Secure, Efficient Sandboxes with System Transactions

Suman Jana          Vitaly Shmatikov
The University of Texas at Austin

Donald E. Porter
Stony Brook University
Untrusted code is everywhere!
Untrusted code is everywhere!
Untrusted code is everywhere!
Sandbox: restrict untrusted code

- Sandbox restricts untrusted code
- Files it can read/write
- System calls and arguments it can use
Properties a sandbox should have

- Uncircumventability
- Fidelity
- Separation policy enforcement and policy specification
- Performance

completely understand effects of untrusted code on host
A quick survey of some sandboxing techniques
Static Analysis

untrusted code

detect malicious code using static-analysis

if benign
execute code

static-analysis is imperfect: false negatives
Incorrect mirroring of system state

Time of check to time of use (TOCTOU) attacks
Building sandboxes with VMs/emulators

- sandbox
  - VM/emulator
    - execute copy of untrusted code
    - if benign
      - execute untrusted code
Building sandboxes with VMs/emulators

sandbox

VM/emulator
execute copy of untrusted code

if (inside VM/Emulator)
  benign actions
else
  malicious actions

if benign
execute untrusted code

split personality attack
Fidelity: necessary for uncircumventability

- Understand behavior of untrusted code
- Semantic gaps can lead to circumvention
- Coherent view of all actions performed by untrusted code
- System calls and arguments
- All affected files (read/write)
How should a sandbox decide which actions to allow/deny?
LEAST PRIVILEGE MODEL

- Whitelist minimal set of operations needed for correct functionality of untrusted code
- Users only have partial information
- Difficult to implement in practice
  - Overestimate: untrusted code can cause more damage
  - Underestimate: crippled functionality
Least privilege model: difficulties

input.avi

untrusted video codec

read ./input.avi

allow read input.avi

write ./out.mpg

allow write out.mpg

sandbox
Least privilege model: difficulties

input.avi
untrusted video codec

read ./input.avi
write ./tmp123
write ./out.mpg

allow read input.avi
allow write out.mpg

sandbox
Least privilege model: difficulties

Untrusted video codec

Read ./input.avi

Write ./tmp123

Write ./out.mpg

Do not know the name a priori

Sandbox

Allow read input.avi

Allow write out.mpg
Least privilege model: difficulties

input.avi

untrusted video codec

read ./input.avi
write ./tmp123
write ./out.mpg

sandbox
allow access to files in current directory

not least privilege anymore
Least privilege model: difficulties

Untrusted video codec

- read ./input.avi
- write ./unrelated
- write ~/bash.rc

Allow access to files in current directory

sandbox

Damage already done

Violation detected and blocked
Recoverability

- Once a sandboxed process is detected doing anything bad, rollback all changes to be safe.

- Real sandboxes have imperfect policies:
  - Can not always enforce least privilege.
  - Only include a subset of possible malicious actions.

- Sandboxes with perfect policies may not need recoverability.
Recoverability can increase parallelism

security checks (e.g. virus scanning)

parallel

sandboxed code
Recoverability can increase parallelism

security checks
(e.g. virus scanning)

parallel sandboxed code

is sandboxed code malicious?
Recoverability can increase parallelism

security checks (e.g. virus scanning)

parallel

is sandboxed code malicious?

yes
Recoverability can increase parallelism

security checks
(e.g. virus scanning)

parallel

is sandboxed code malicious?

yes

sandboxed code

recover
Properties a sandbox should have

- Uncircumventability
- Separation policy enforcement and policy specification
- Performance
- Recoverability
**OS Transactions**

speculative execution

![Diagram showing speculative execution in an operating system with files A and B involved.](image)
OS transactions

speculative execution

system call

file A

file B
OS transactions

speculative execution

system call

modified
file A

transactional
work-set

OS

file A

file B
OS transactions

speculative execution

system call

abort/commit

modified file A
modified file B

transactional work-set

OS

file A
file B
OS transactions

speculative execution

commit

make changes visible to other processes
OS transactions

speculative execution

commit

file A

file B

make changes visible to other processes
Security needs transactions

speculatively execute untrusted code

transactional work-set

policy violation?

commit

no

abort

yes
Security needs transactions

speculatively execute untrusted code

performance (no blocking)

transactional work-set

policy violation?

uncircumventability

recoverability

no commit

yes abort
OS support for transactions

- TxOS: Porter et al. SOSP 2009
- Speculative execution support for 150+ system calls
- Provides ACID semantics
- Originally done for handling concurrency
Insight: transactions are great match for security
Execute untrusted code inside a transaction
Make security decisions by checking work-set
Parallelize security checks with program execution
Abort transaction if anything malicious is detected
Insight: transactions are great match for security

- Execute untrusted code inside a transaction
- Make security decisions by checking work-set
- Parallelize security checks with program execution
- Abort transaction if anything malicious is detected
Evaluation

- Can TxBox isolate large real-world programs?
  - FFmpeg: audio/video codec
  - SpiderMonkey: JavaScript engine
  - Vim: editor

- How much performance/memory overhead does TxBox incur?
On average TxBox causes less than < 20% runtime overhead compared to Linux
TxBox: memory overhead

On average TxBox execution of a process takes 2x more memory compared to regular Linux execution.
ClamAV “on-open” scan
**TxBox parallelization gain**

*(ClamAV scanning)*

230 %

host had
4 cores
Conclusion: security needs transactions

- Speculatively execute untrusted code
- Rollback if any malice is detected
- Inspect all effects of the untrusted process at the right level of abstraction
- Prevent circumvention and evasion
Conclusion: security needs transactions

- Speculatively execute untrusted code
- Rollback if any malice is detected
- Inspect all effects of the untrusted process at the right level of abstraction
- Prevent circumvention and evasion

suman@cs.utexas.edu
Recoverability: output commit problem

- How to maintain recoverability if an untrusted process performs network i/o?

- Unsolvable in general, we do the next best thing

- Always preserve local recoverability

  - Deny network i/o and continue

  - Execute network i/o outside of transaction and continue
**TxBox: Implementation Issues**

- TxOS transactions need cooperative processes calling
  - \( \texttt{xbegin} \)
  - \( \texttt{xend} \)
- Untrusted processes are not co-operative
- Support “forced” transactions
- Implement policy manager and policy enforcer
- See paper for details
Building sandboxes with system call interposition

sandboxed code
syscall args

system call

user-space

kernel

system call wrapper

allow/deny

should allow?
Building sandboxes with system call interposition

sandboxed code
syscall args

system call
should allow?

if allowed
system call wrapper

allow/deny

system call specific handler

kernel

user-space
Building sandboxes with system call interposition

sandboxed code
syscall args

system call wrapper

system call

kernel

user-space

sandbox

allow/deny

possible race condition

should allow?

if allowed

system call specific handler
Time of check to time of use (TOCTOU) attacks

sandboxed code

syscall args

fork

untrusted code (child)

user-land

system call

kernel

system call wrapper
**Time of check to time of use (TOCTOU) attacks**

- Sandbox code
- Syscall args
- Untrusted code (child)
- Sandbox
- Fork
- System call
- Should allow?
- Kernel
- User-land
- System call wrapper

**Legend:**
- Sandboxed code
- System call wrapper
**Time of check to time of use (TOCTOU) attacks**

- **sandboxed code**
  - syscall args

- **untrusted code (child)**
  - overwrite

- **sandbox**

- **user-land**
  - system call

- **kernel**
  - should allow?

- **system call wrapper**
  - allow
  - system call specific handler
**Time of check to time of use (TOCTOU) attacks**

- **sandboxed code**
  - syscall args

- **untrusted code (child)**
  - fork
  - overwrite

- **system call wrapper**
  - system call
  - should allow?

- **sandbox**
  - allow

- **kernel**
  - user-land