TxBox: Building Secure, Efficient Sandboxes with System Transactions

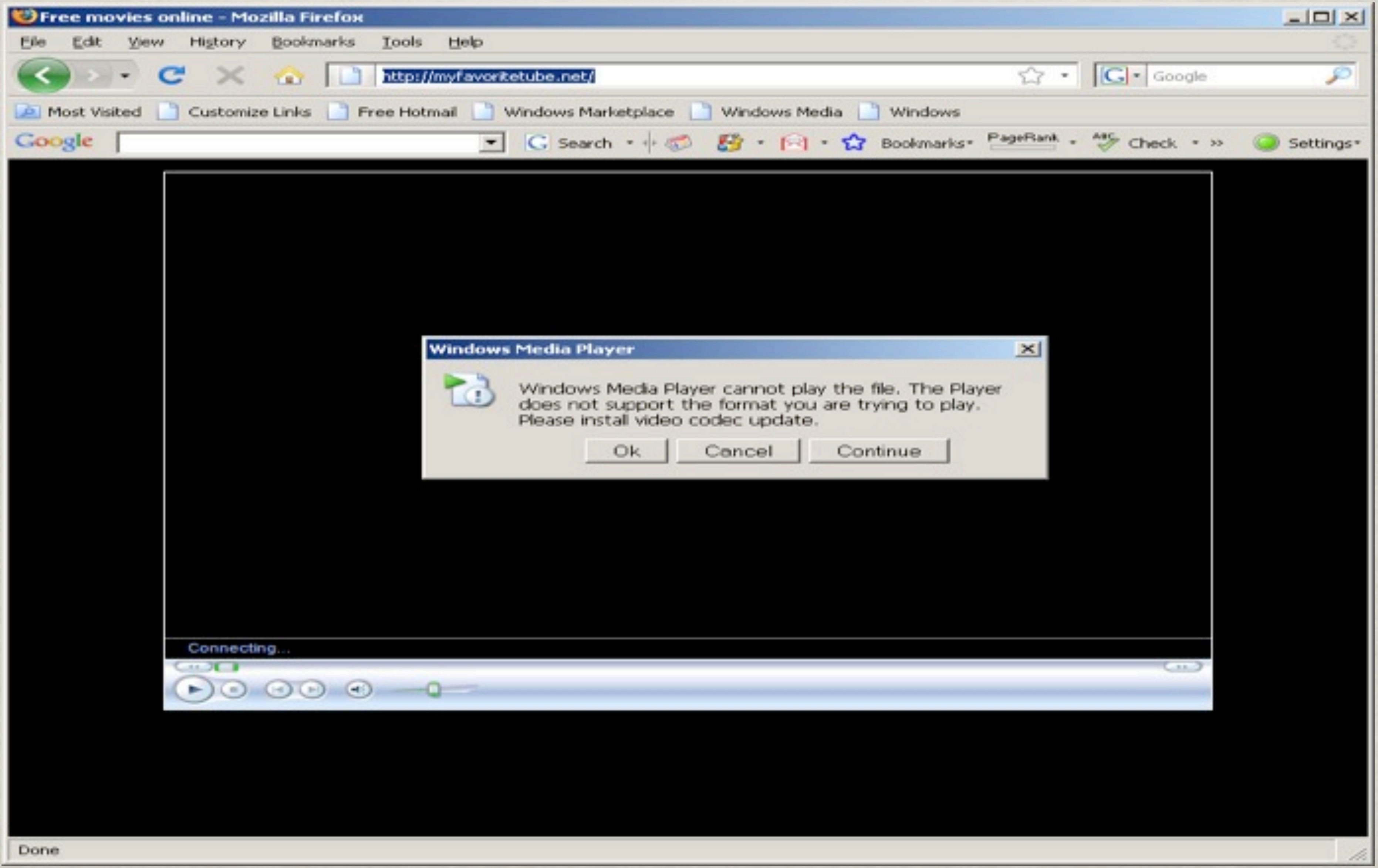
Suman Jana **Vitaly Shmatikov**

The University of Texas at Austin

Donald E. Porter

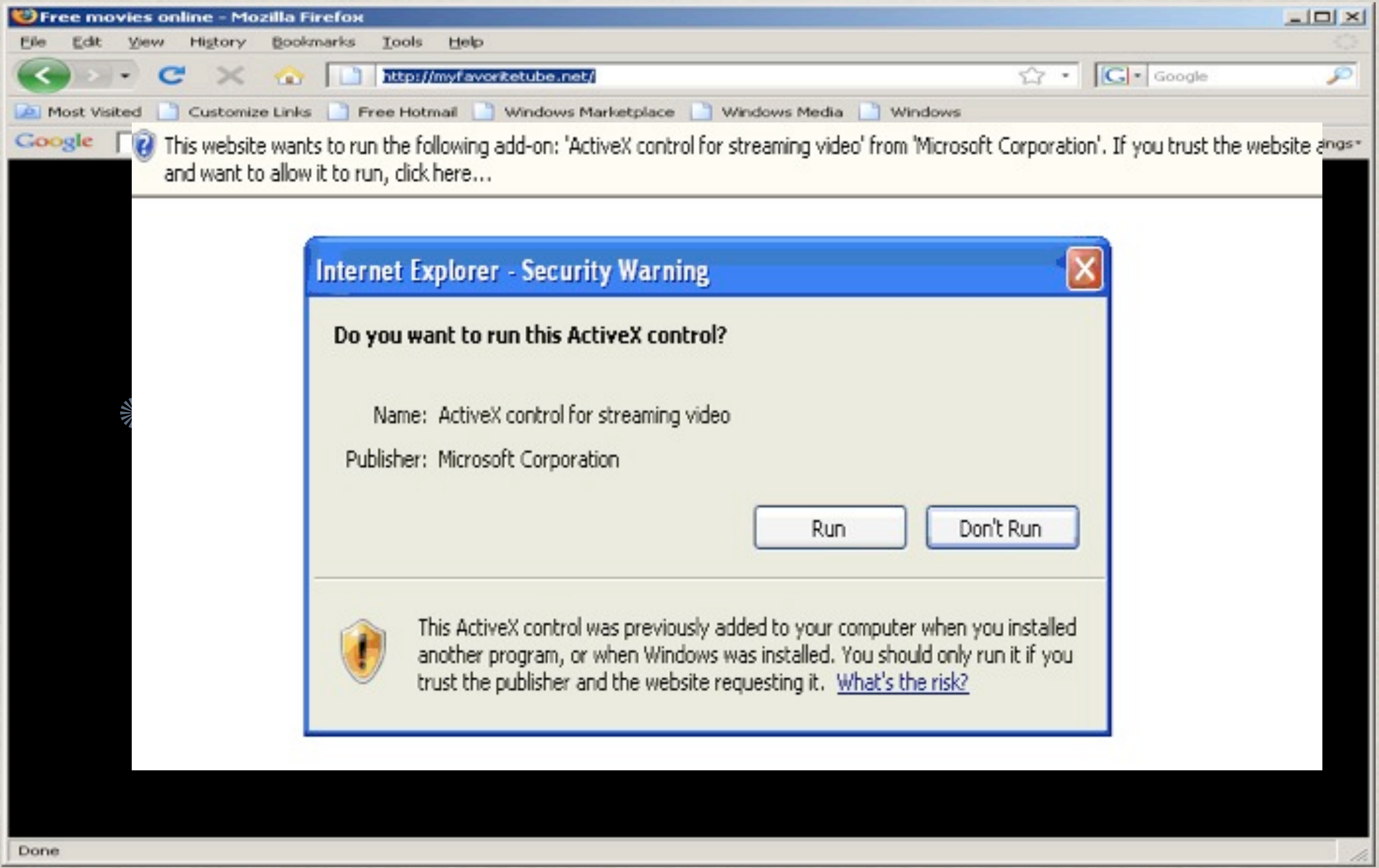
Stony Brook University

Untrusted code is everywhere !



Windows Media Player cannot play the file. The Player does not support the format you are trying to play. Please install video codec update.

Untrusted code is everywhere !



This website wants to run the following add-on: 'ActiveX control for streaming video' from 'Microsoft Corporation'. If you trust the website and want to allow it to run, click here...

Internet Explorer - Security Warning

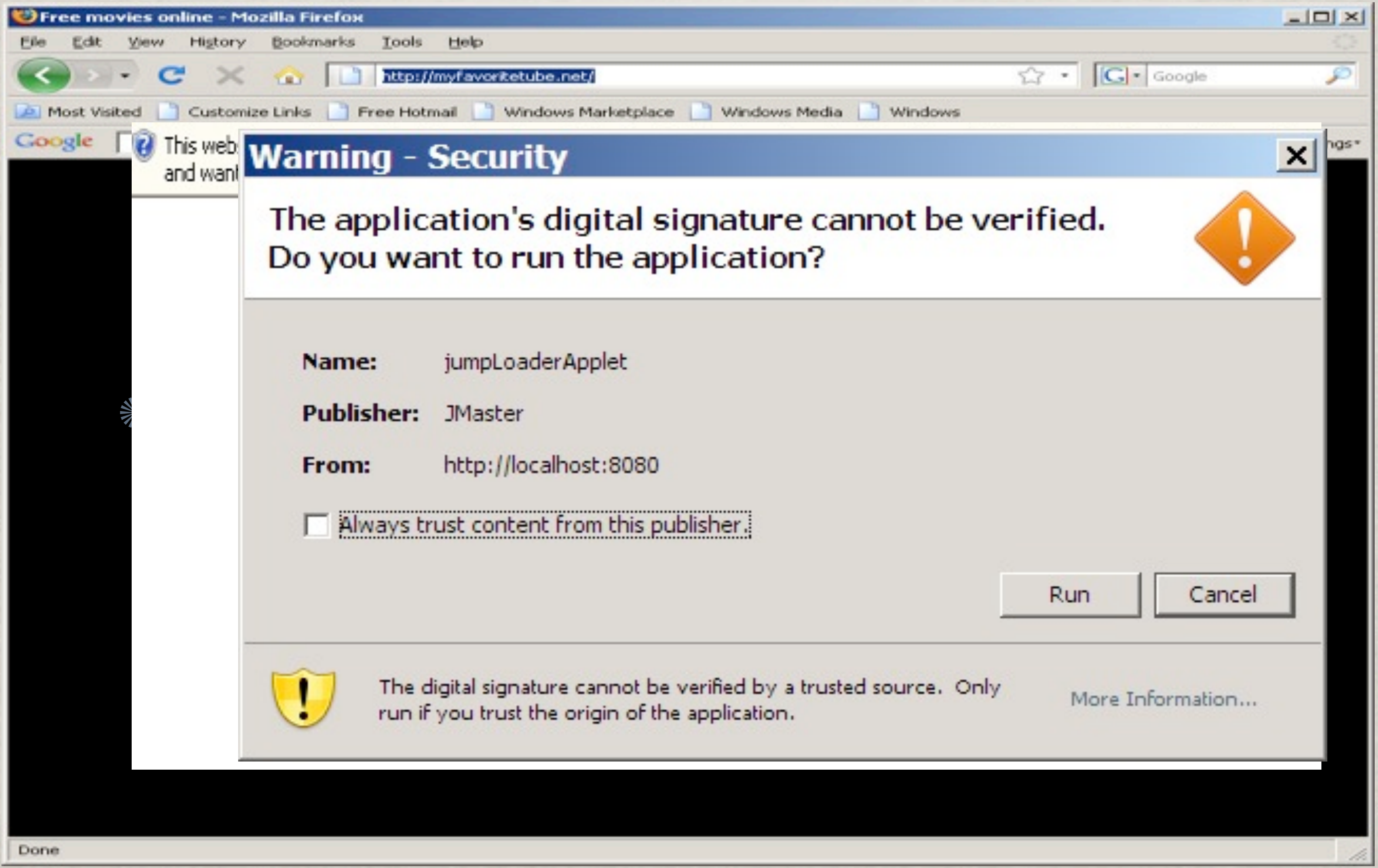
Do you want to run this ActiveX control?

Name: ActiveX control for streaming video

Publisher: Microsoft Corporation

This ActiveX control was previously added to your computer when you installed another program, or when Windows was installed. You should only run it if you trust the publisher and the website requesting it. What's the risk?

Untrusted code is everywhere !



Free movies online - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://myfavoritetube.net/
Google
Most Visited Customize Links Free Hotmail Windows Marketplace Windows Media Windows
This web
and want
Done

Warning - Security

The application's digital signature cannot be verified.
Do you want to run the application?

Name: jumpLoaderApplet

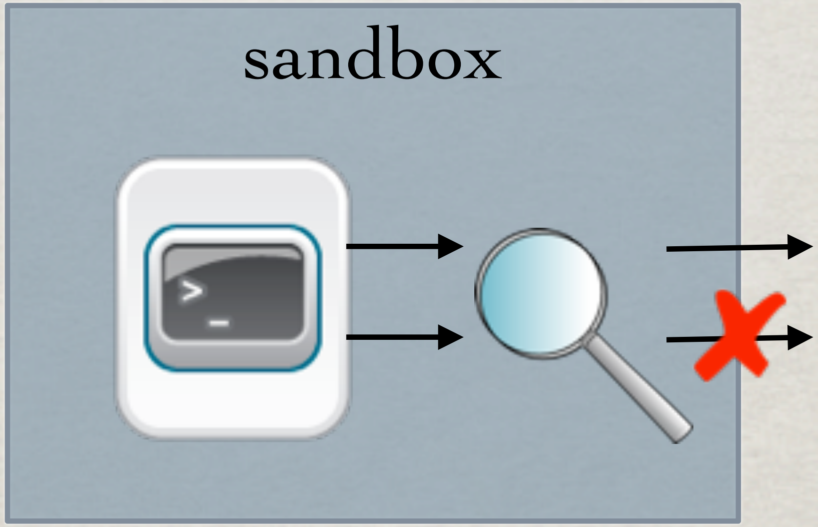
Publisher: JMaster

From: http://localhost:8080

□ Always trust content from this publisher.

The digital signature cannot be verified by a trusted source. Only run if you trust the origin of the application.

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

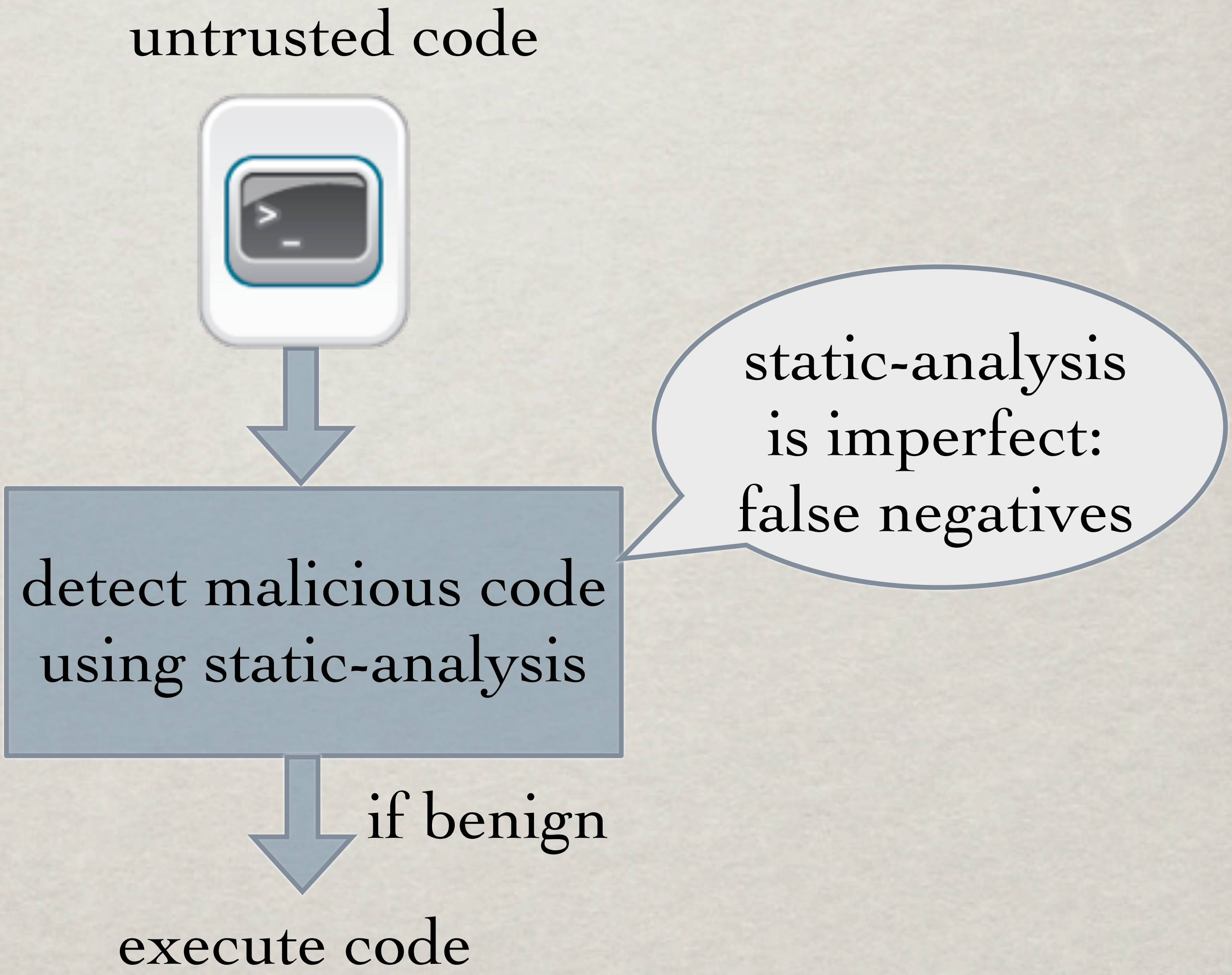
- Uncircumventability
  - Fidelity

completely understand effects of untrusted code on host

- Separation policy enforcement and policy specification
- Performance

A quick survey of some sandboxing techniques

Static Analysis



untrusted code

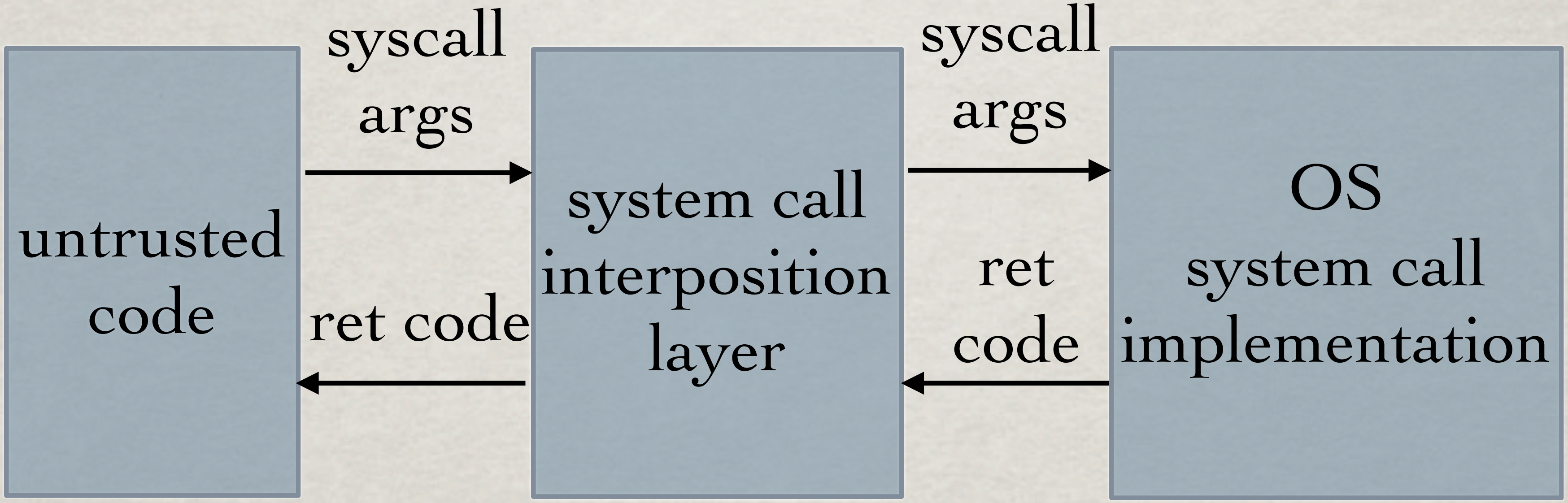
detect malicious code using static-analysis

if benign

execute code

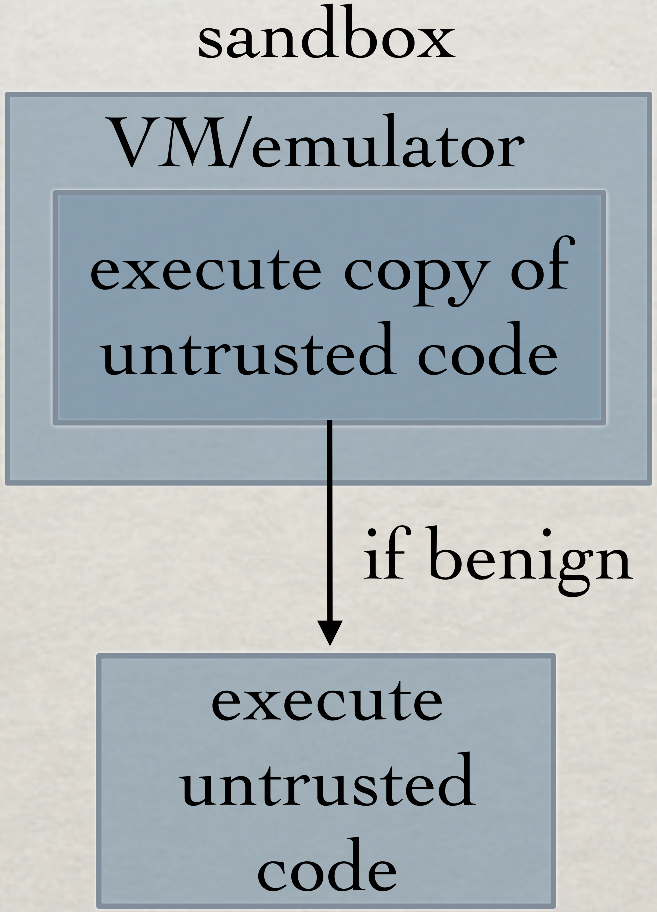
static-analysis is imperfect: false negatives

System call interposition


untrusted code
syscall args
ret code
system call interposition layer
syscall args
ret code
OS system call implementation

- Incorrect mirroring of system state
- Time of check to time of use (TOCTOU) attacks

Building sandboxes with VMs/emulators



sandbox

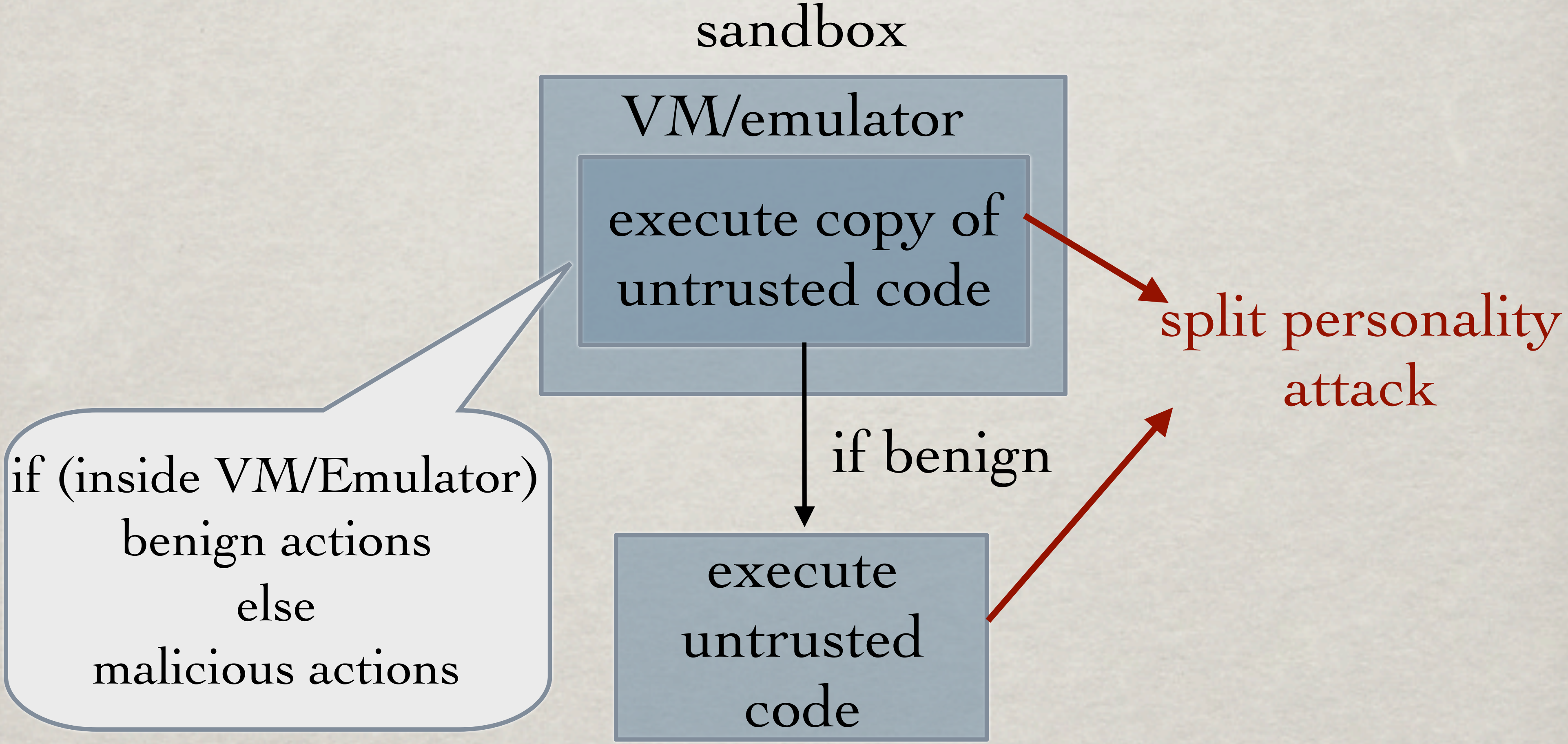
VM/emulator

execute copy of untrusted code

if benign

execute untrusted code

Building sandboxes with VMs/emulators



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)

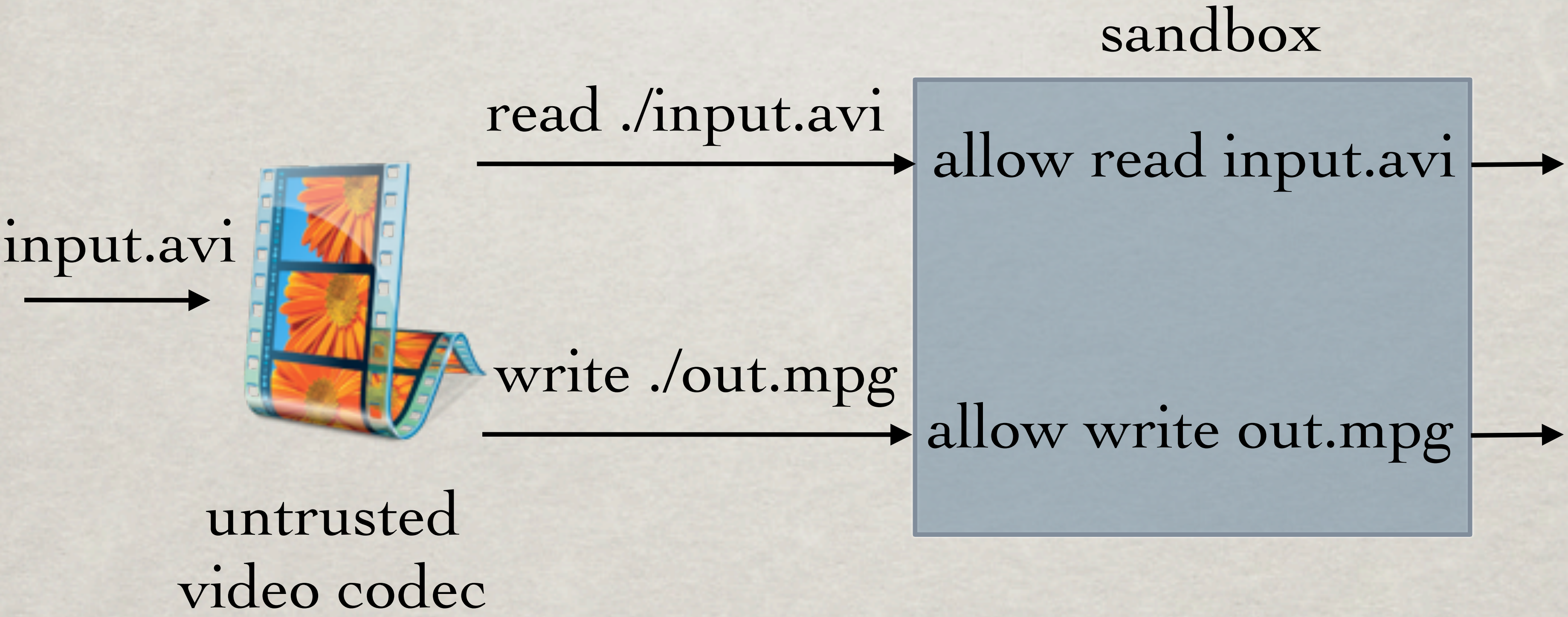
Sandbox policies

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



sandbox

read ./input.avi

allow read input.avi

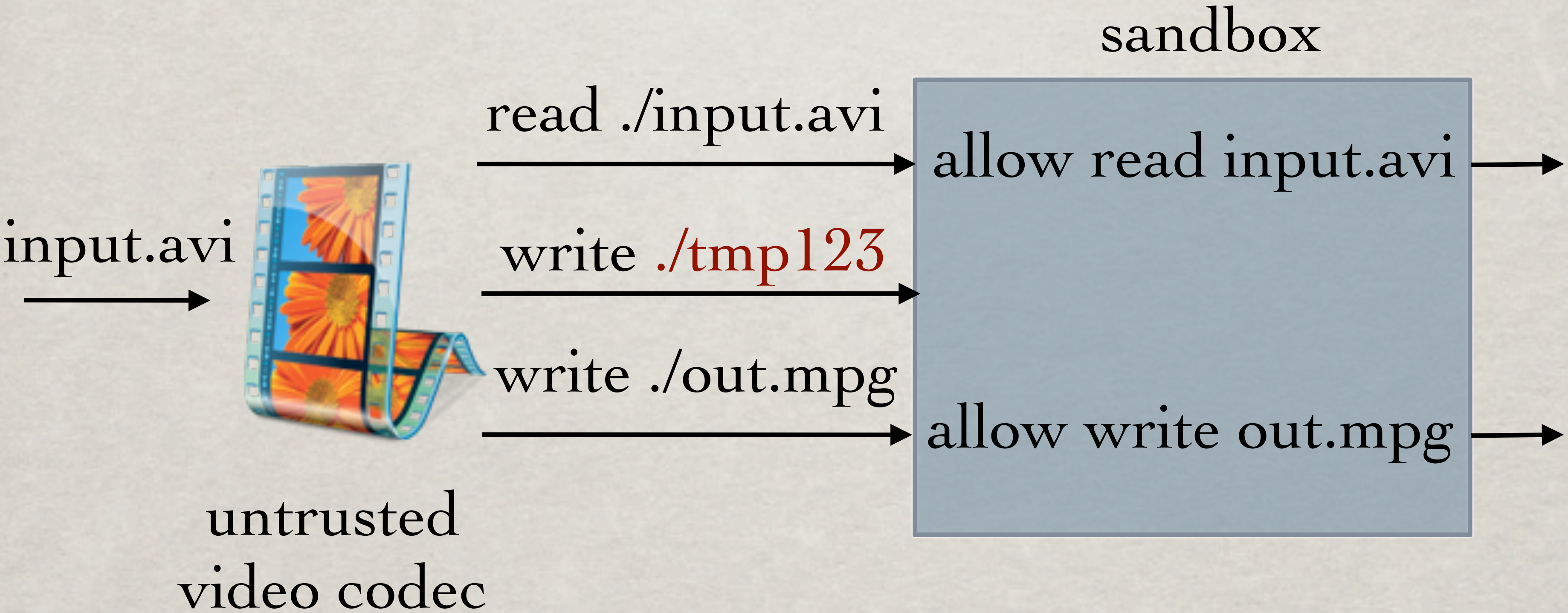
input.avi

write ./out.mpg

allow write out.mpg

untrusted
video codec

Least privilege model: difficulties



sandbox

input.avi

read ./input.avi

write ./tmp123

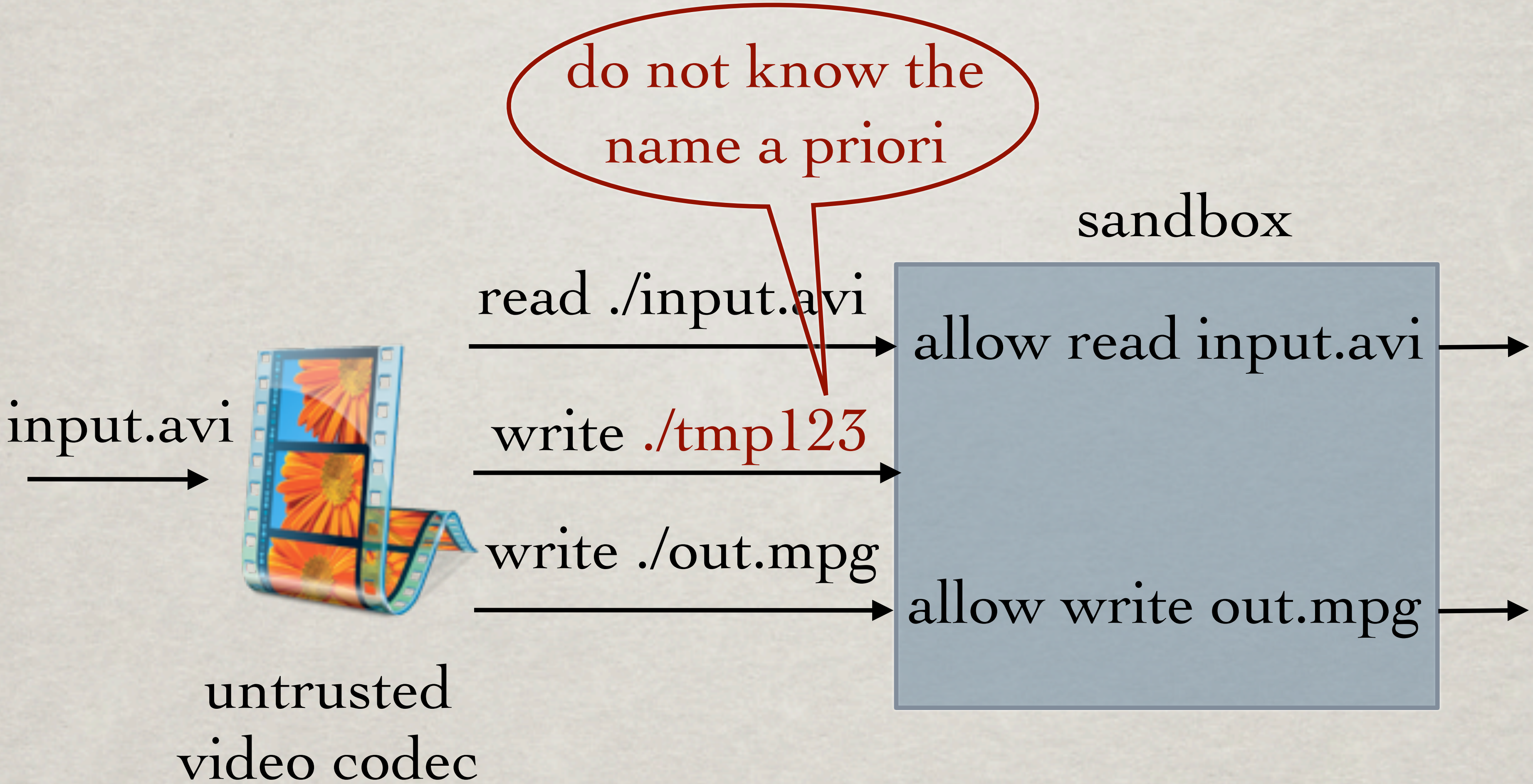
write ./out.mpg

allow read input.avi

allow write out.mpg

untrusted video codec

Least privilege model: difficulties



do not know the name a priori

sandbox

input.avi

read ./input.avi

allow read input.avi

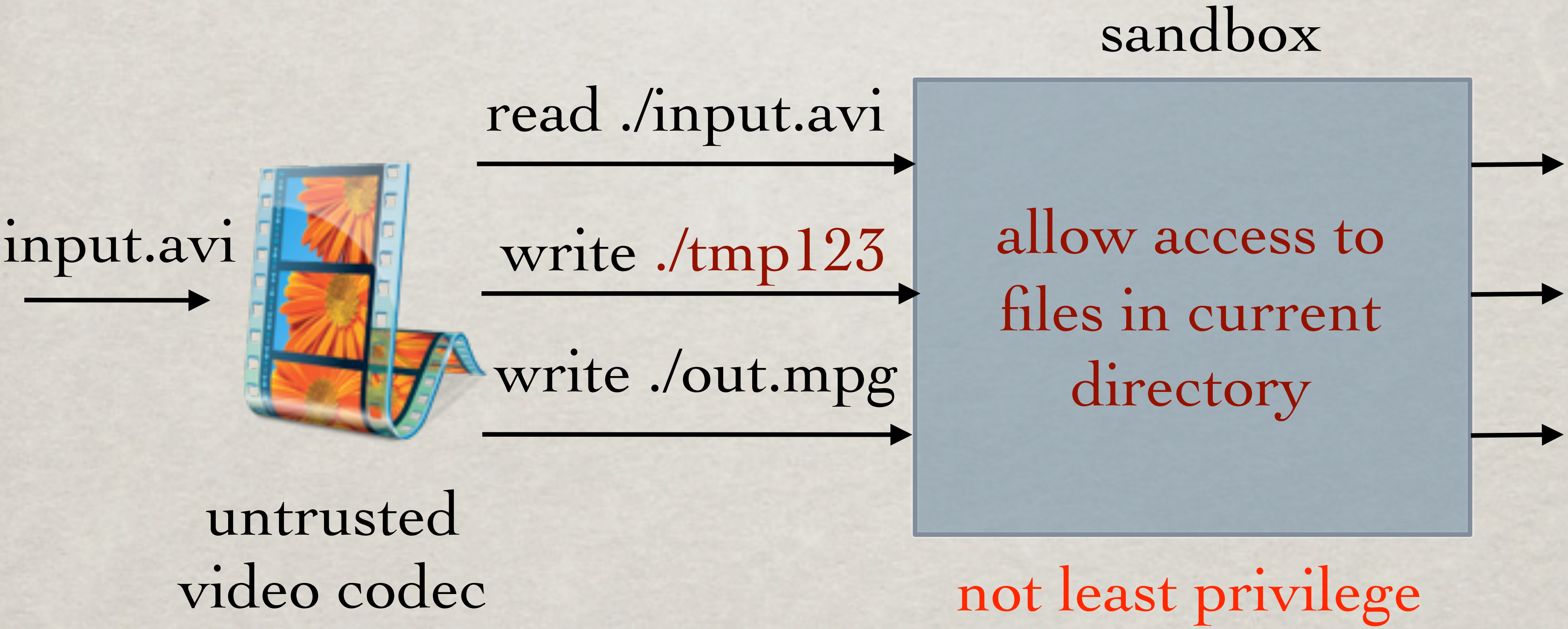
write ./tmp123

write ./out.mpg

allow write out.mpg

untrusted video codec

Least privilege model: difficulties



sandbox

input.avi

read ./input.avi

write ./tmp123

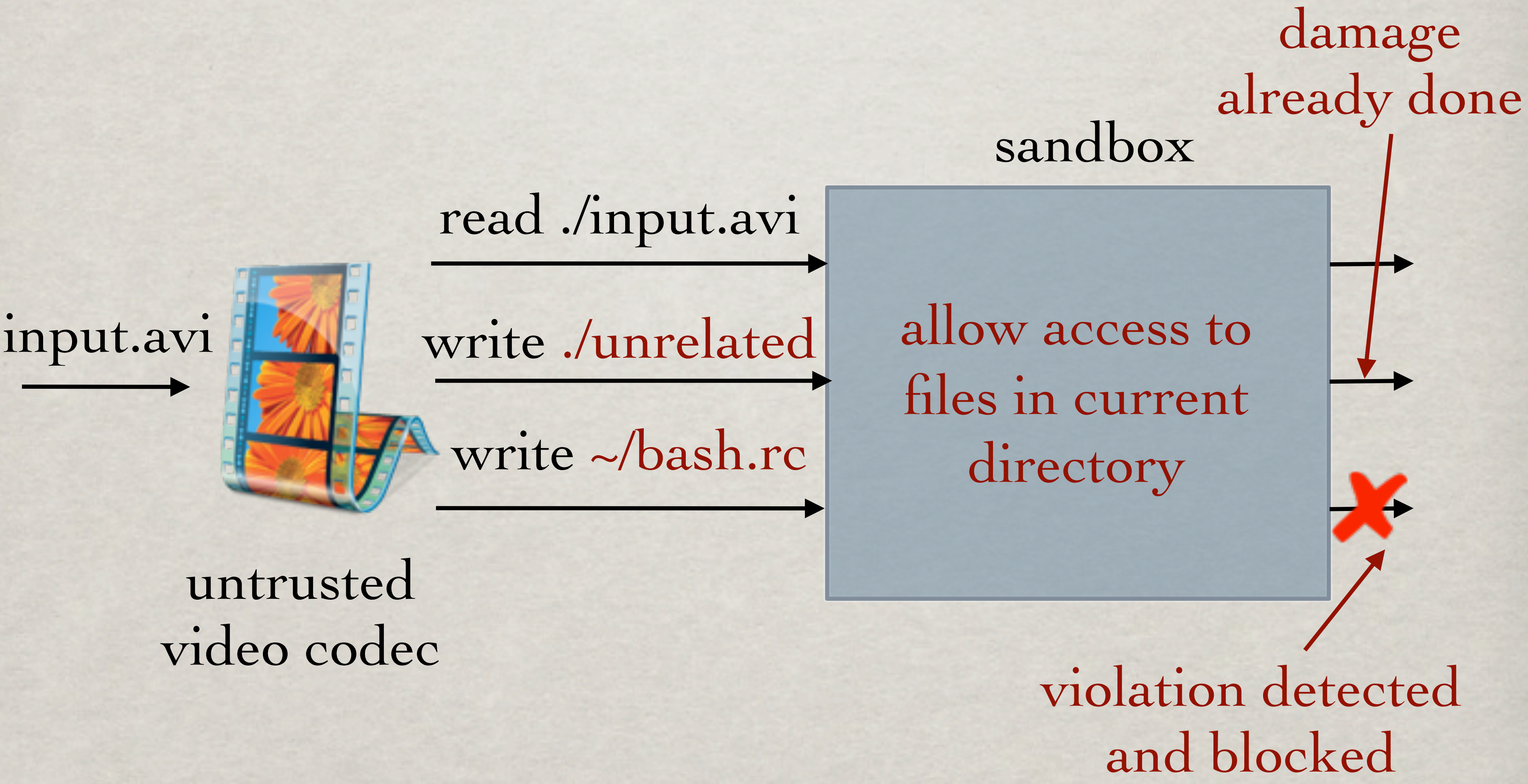
write ./out.mpg

untrusted video codec

allow access to files in current directory

not least privilege anymore

Least privilege model: difficulties



input.avi

untrusted video codec

read ./input.avi

write ./unrelated

write ~/bash.rc

sandbox

allow access to files in current directory

damage already done

violation detected and blocked

Recoverability

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

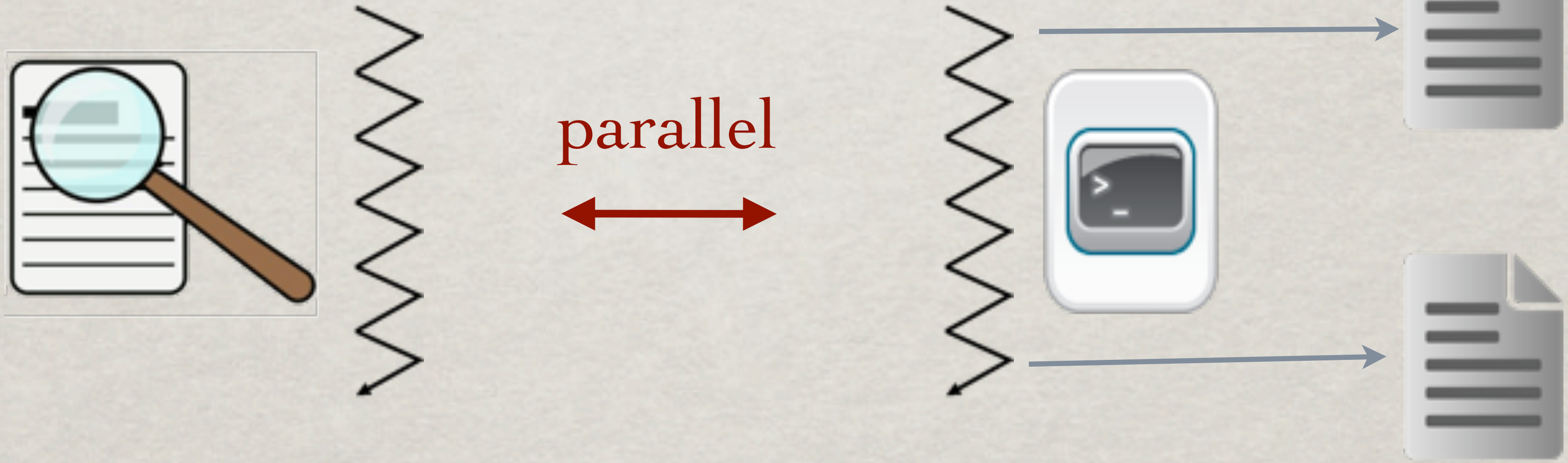
- Real sandboxes have imperfect policies

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

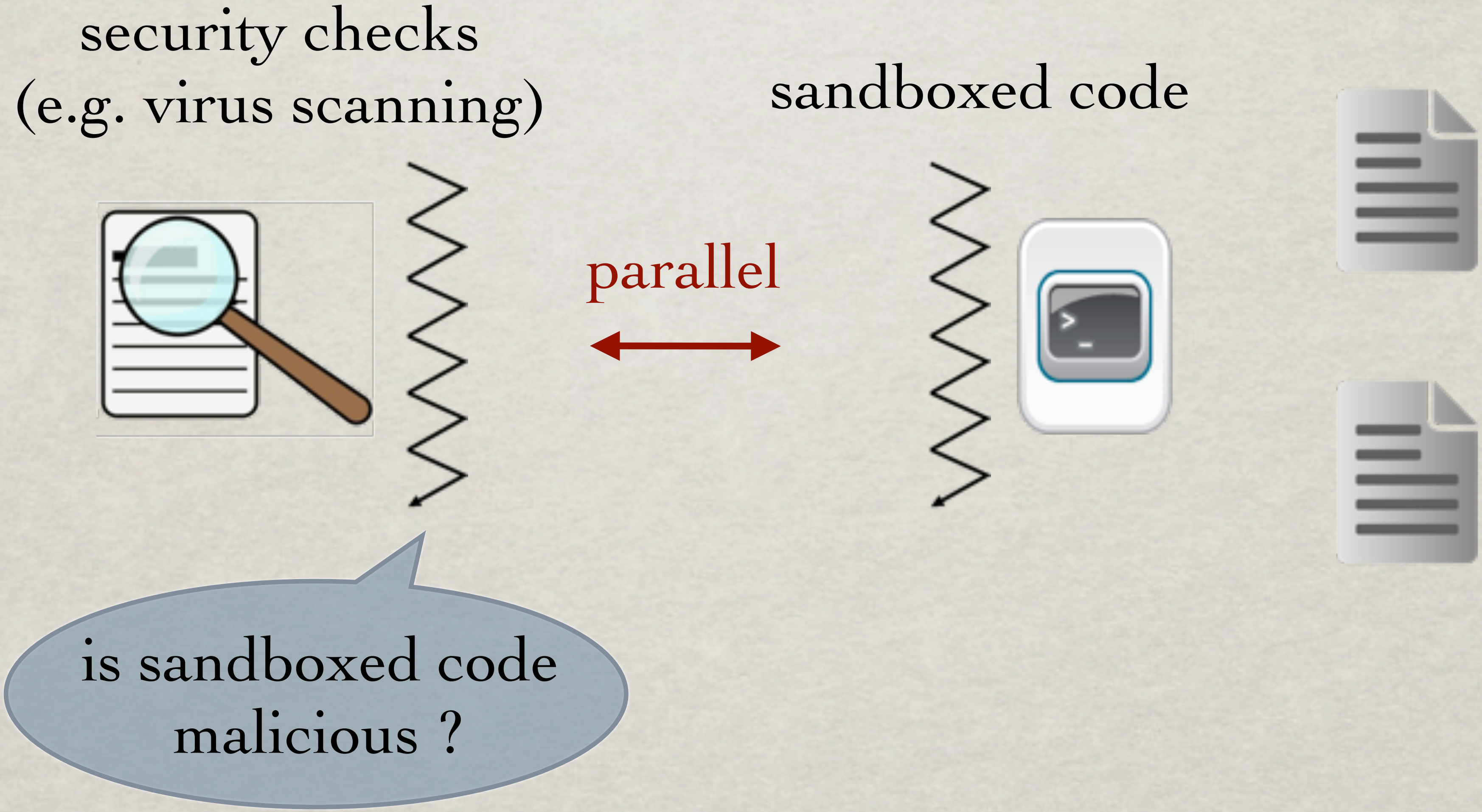


security checks
(e.g. virus scanning)

parallel

sandboxed code

Recoverability can increase parallelism



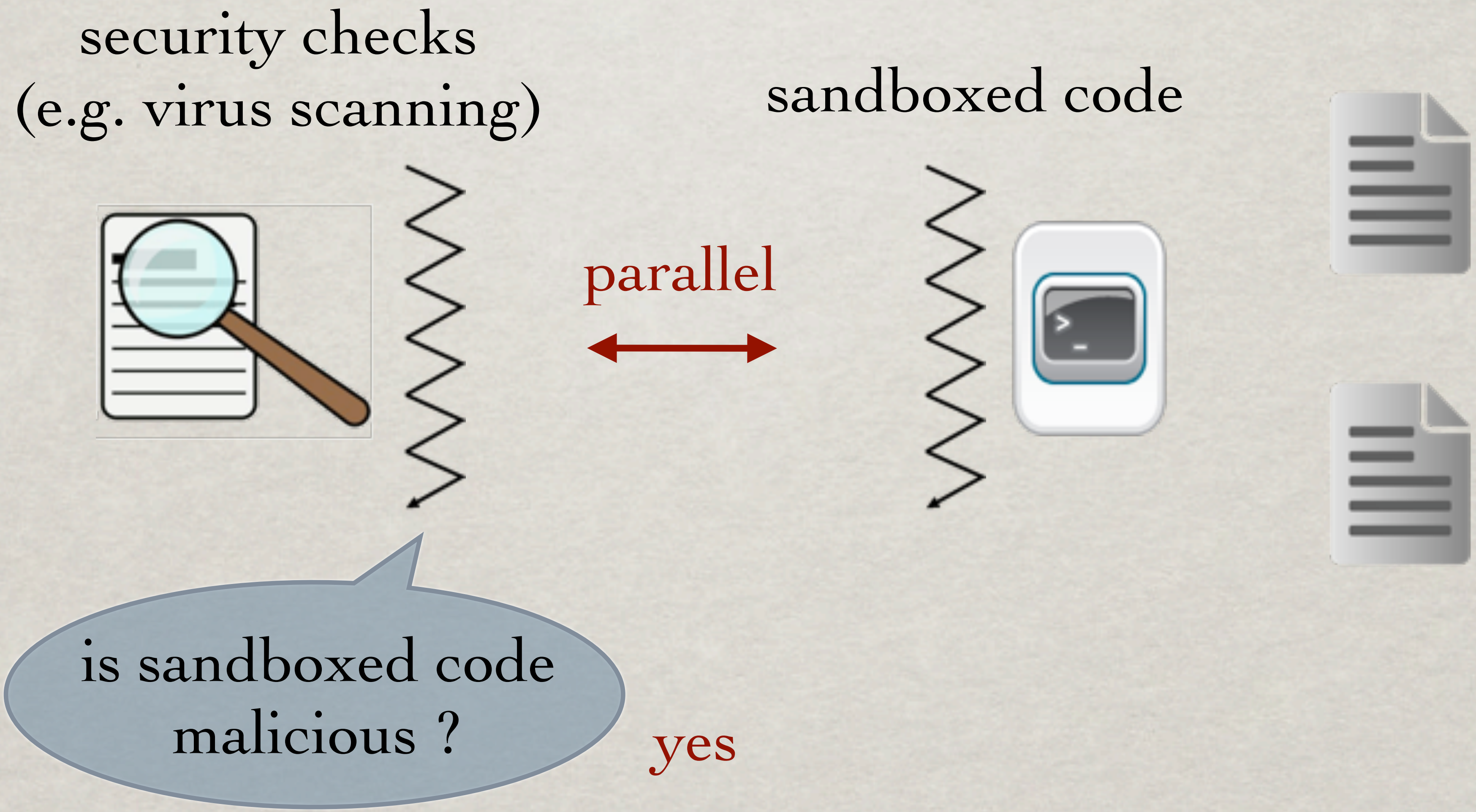
security checks (e.g. virus scanning)

parallel

sandboxed code

is sandboxed code malicious ?

Recoverability can increase parallelism



security checks (e.g. virus scanning)

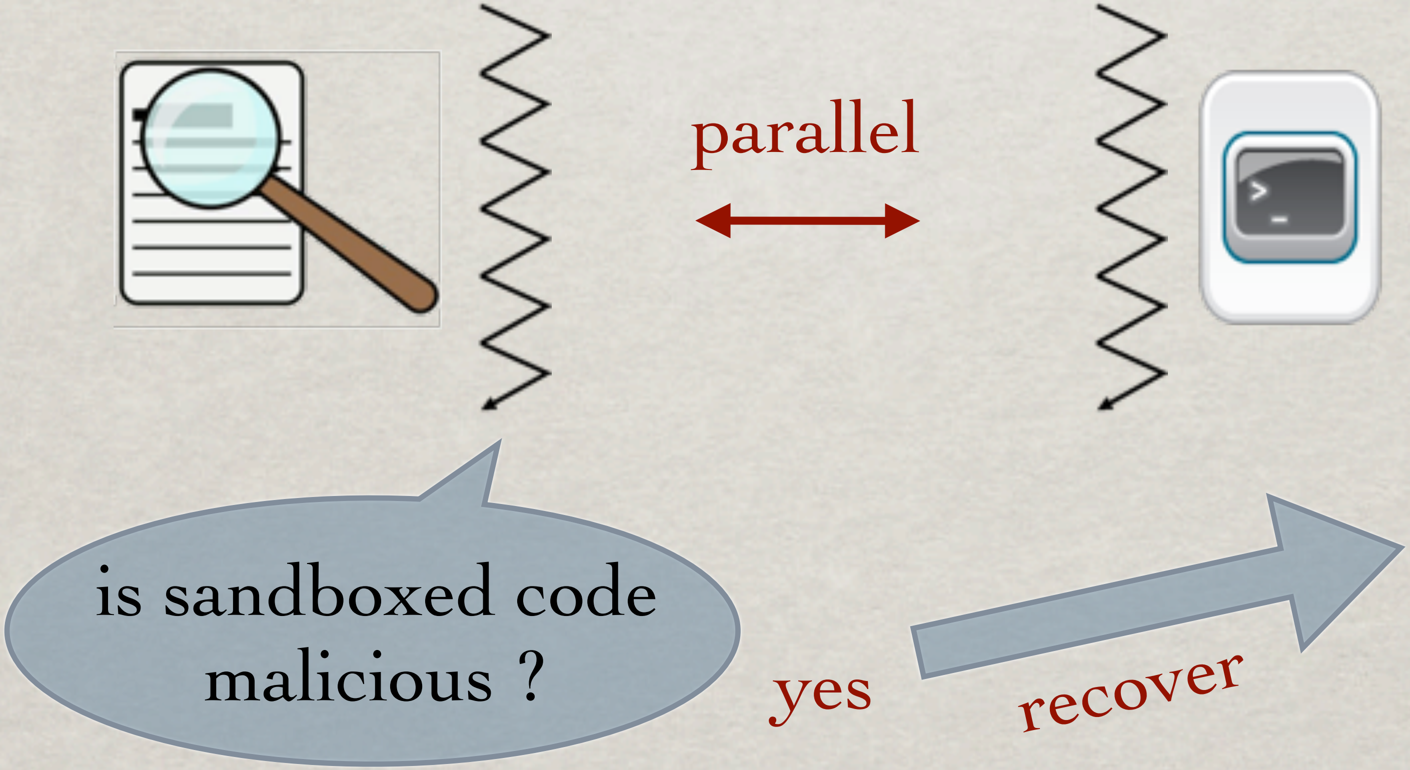
sandboxed code

parallel

is sandboxed code malicious ?

yes

Recoverability can increase parallelism



security checks
(e.g. virus scanning)

sandboxed code

parallel

is sandboxed code malicious ?

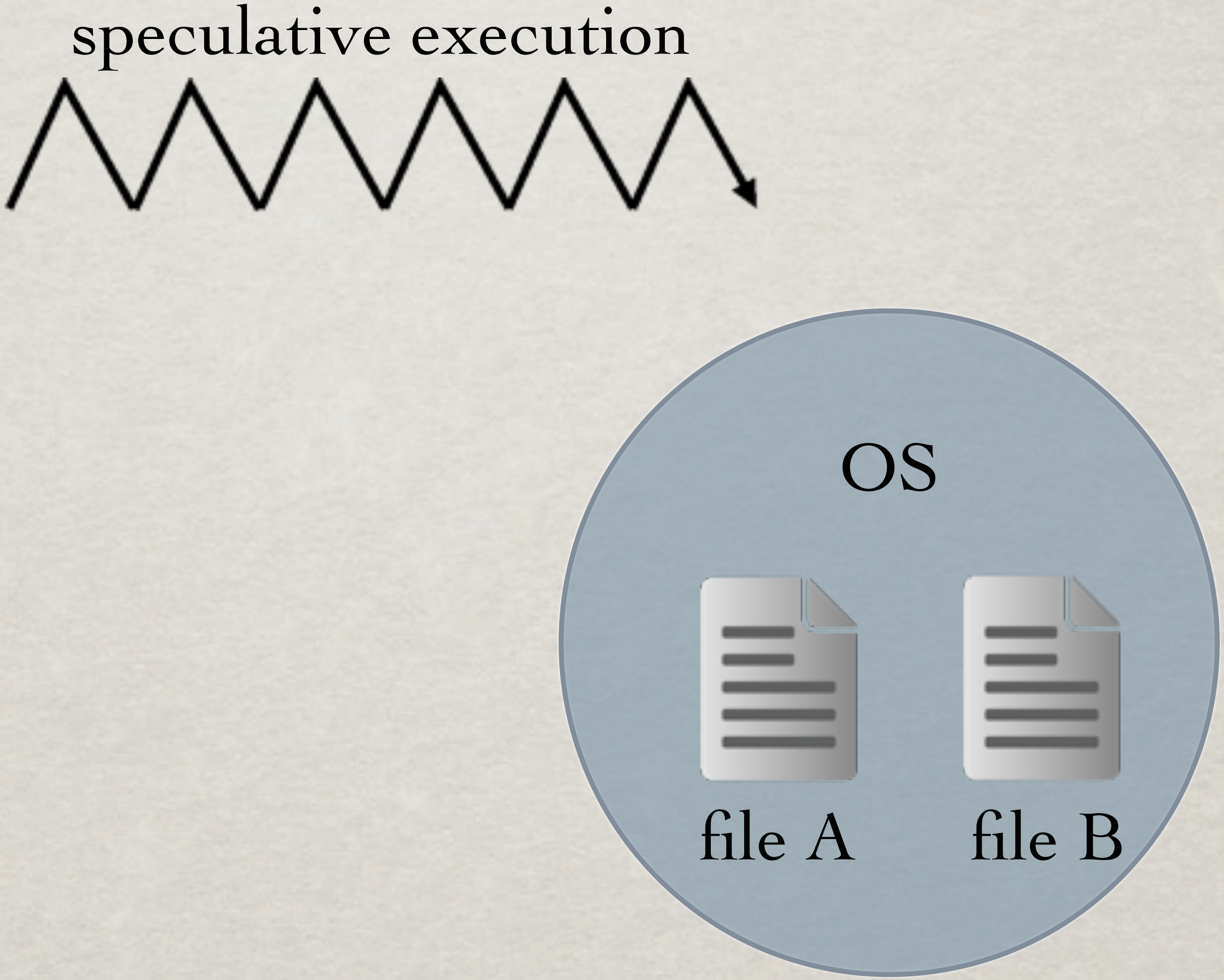
yes

recover

Properties a sandbox should have

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

OS transactions



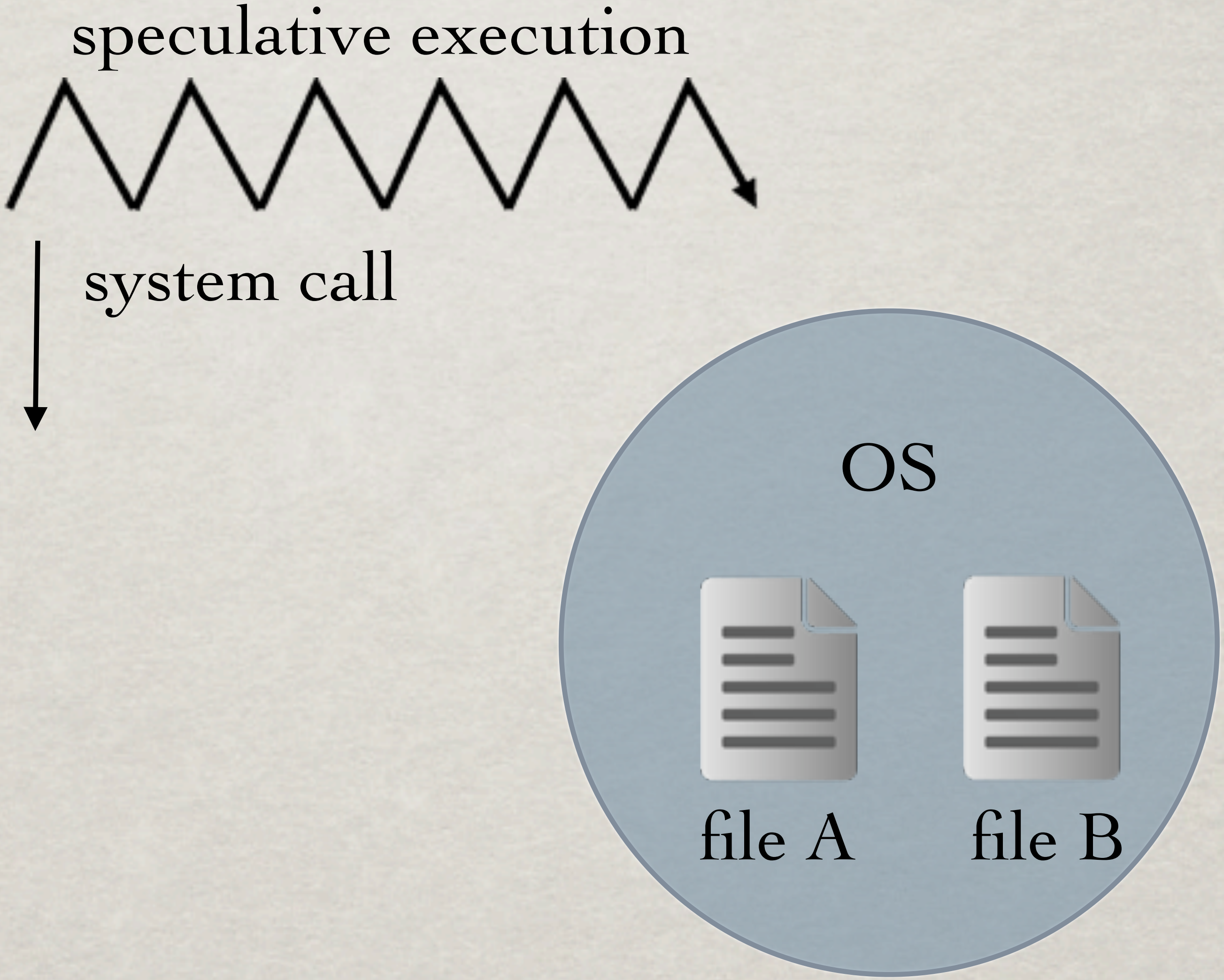
speculative execution

OS

file A

file B

OS transactions



speculative execution

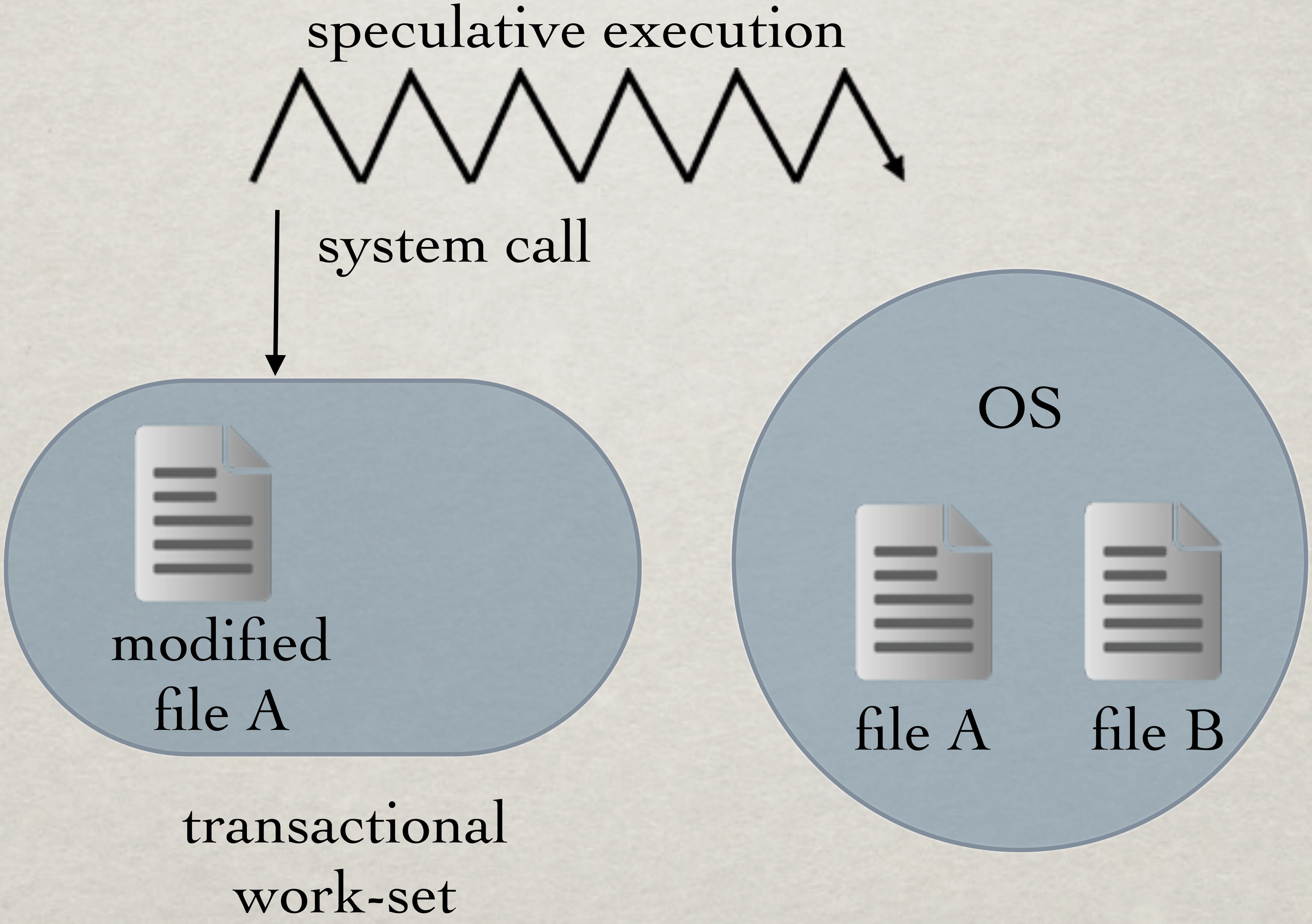
system call

OS

file A

file B

OS transactions


speculative execution

system call

modified file A

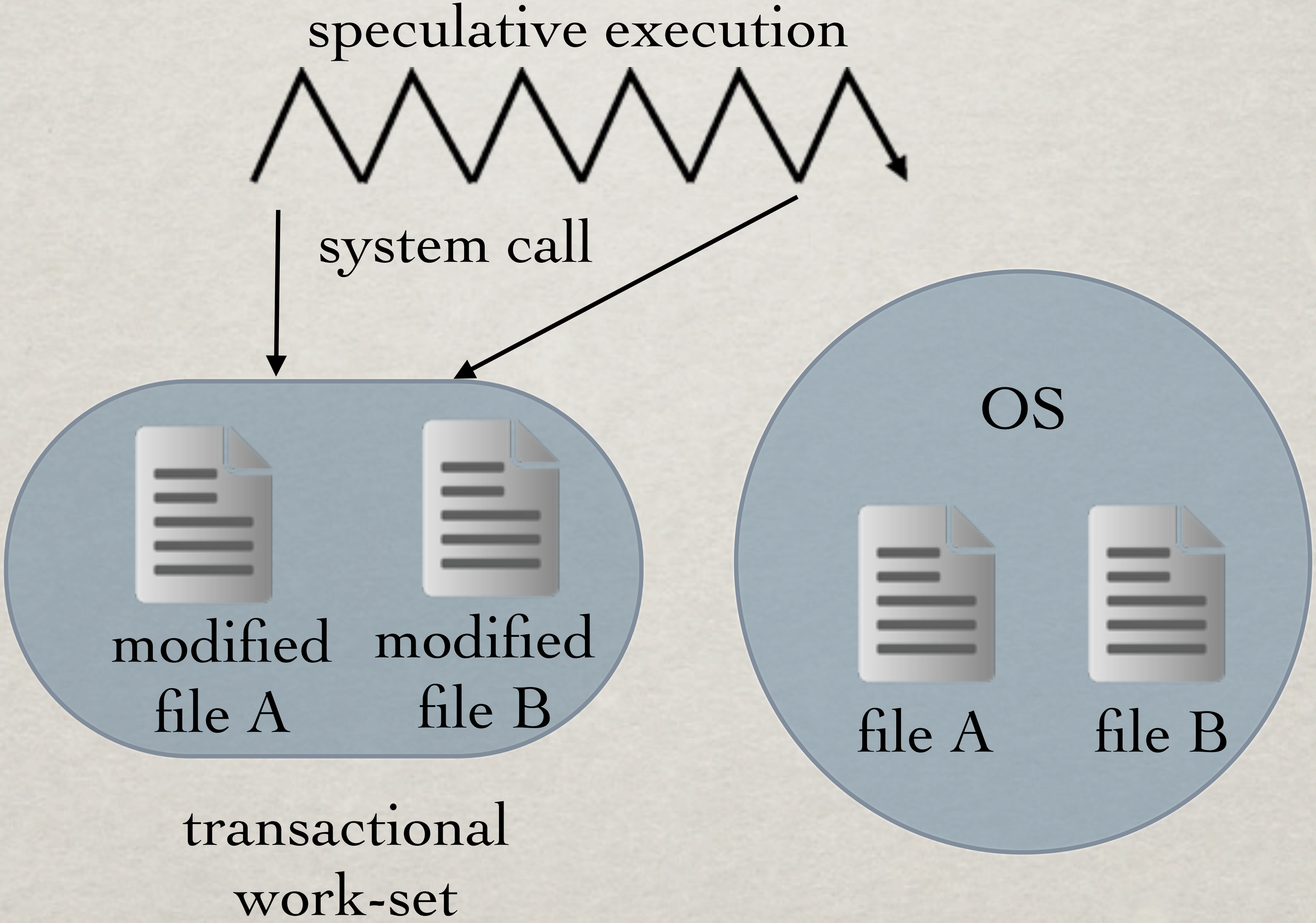
transactional work-set

OS

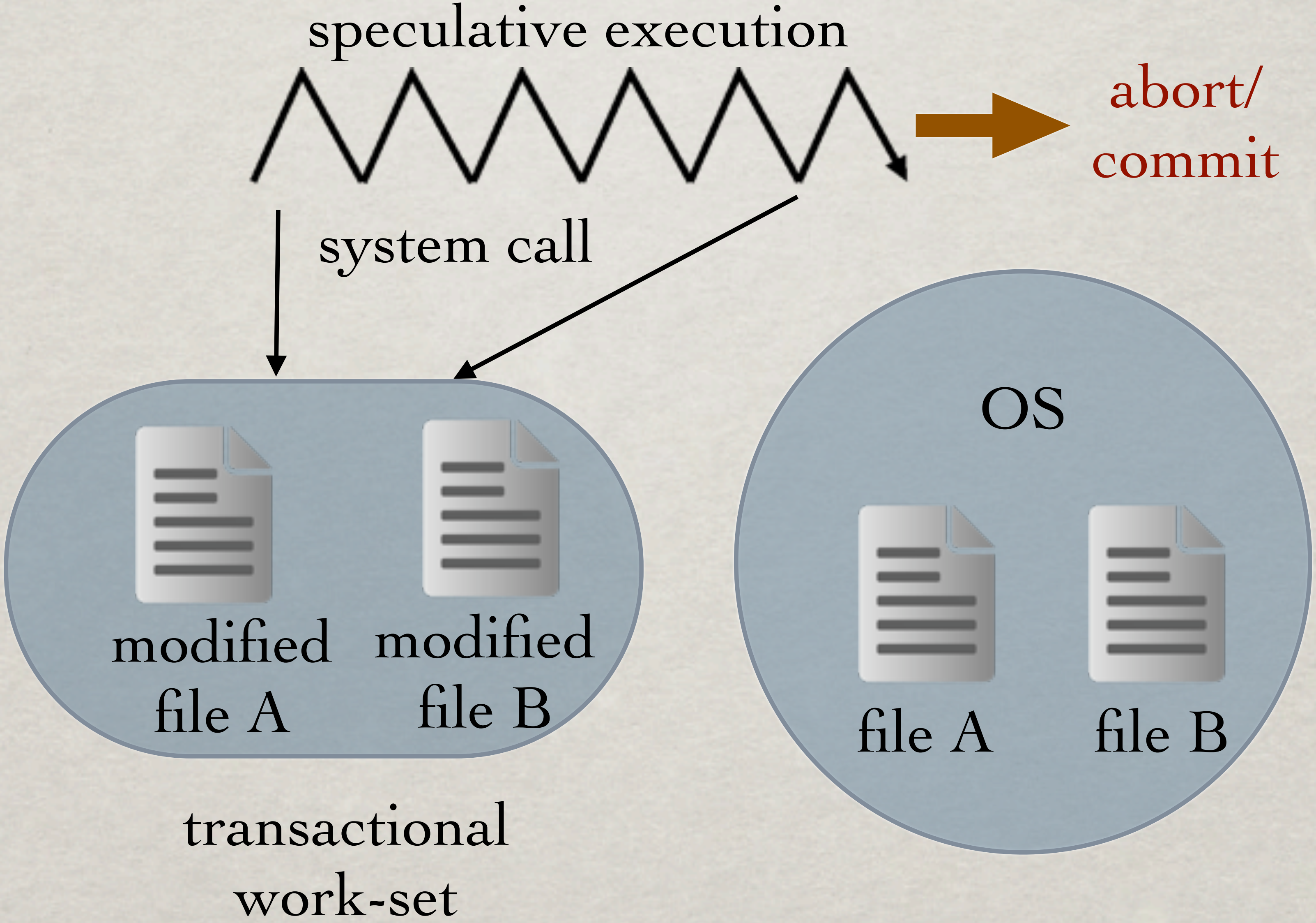
file A

file B

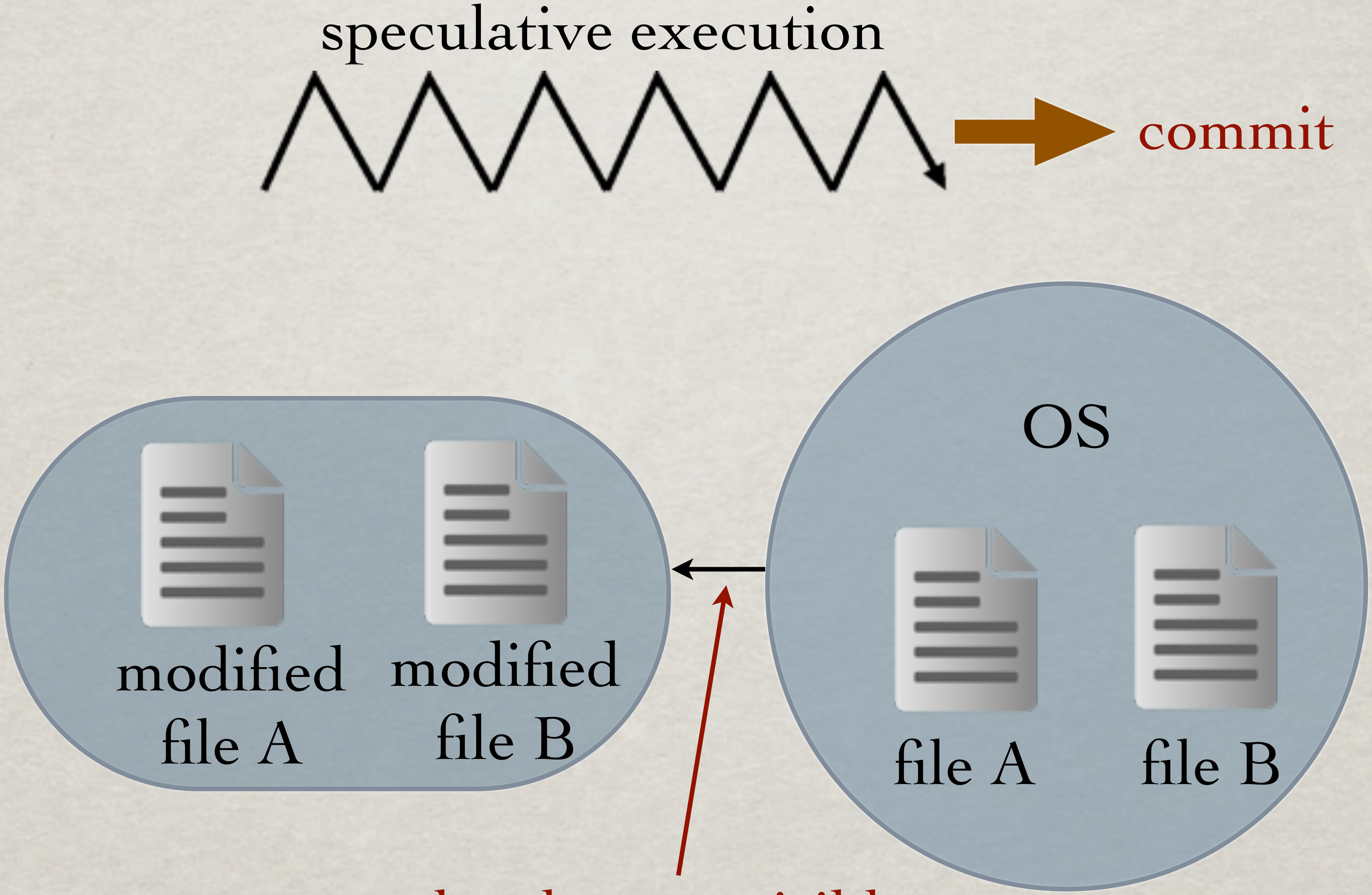
OS transactions



OS transactions

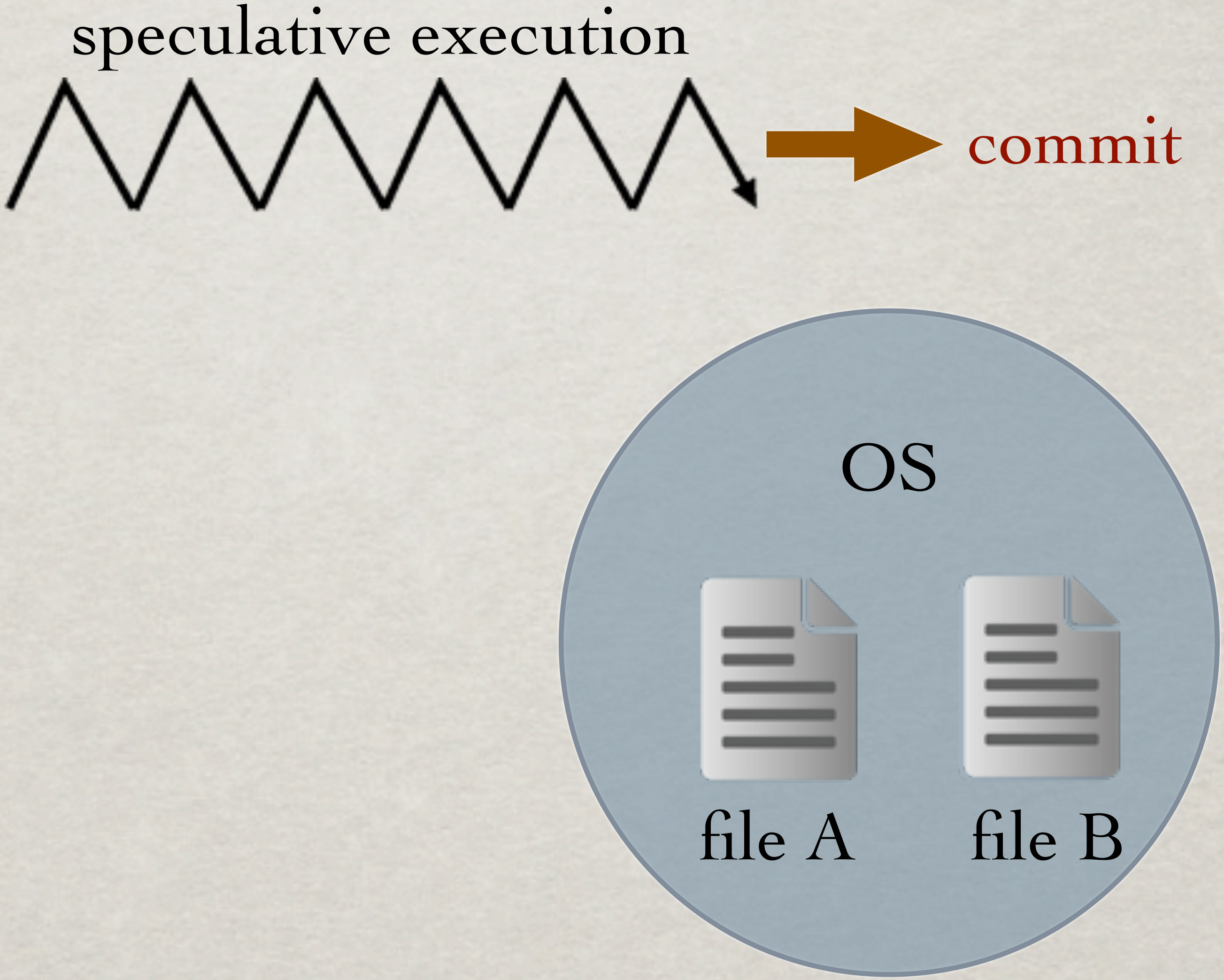


OS transactions



make changes visible
to other processes

OS transactions



speculative execution

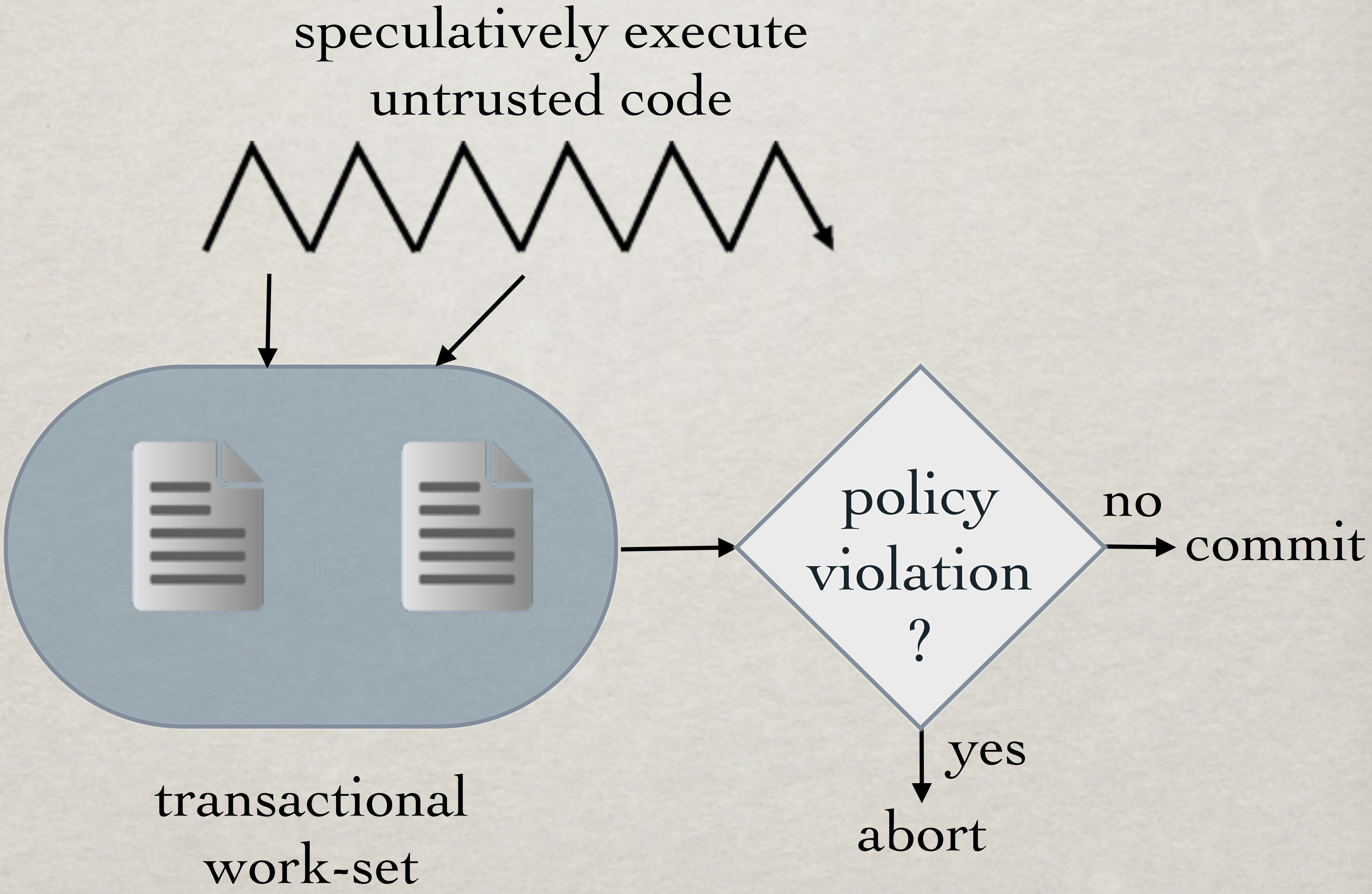
commit

OS

file A

file B

Security needs transactions



speculatively execute untrusted code

transactional work-set

policy violation ?

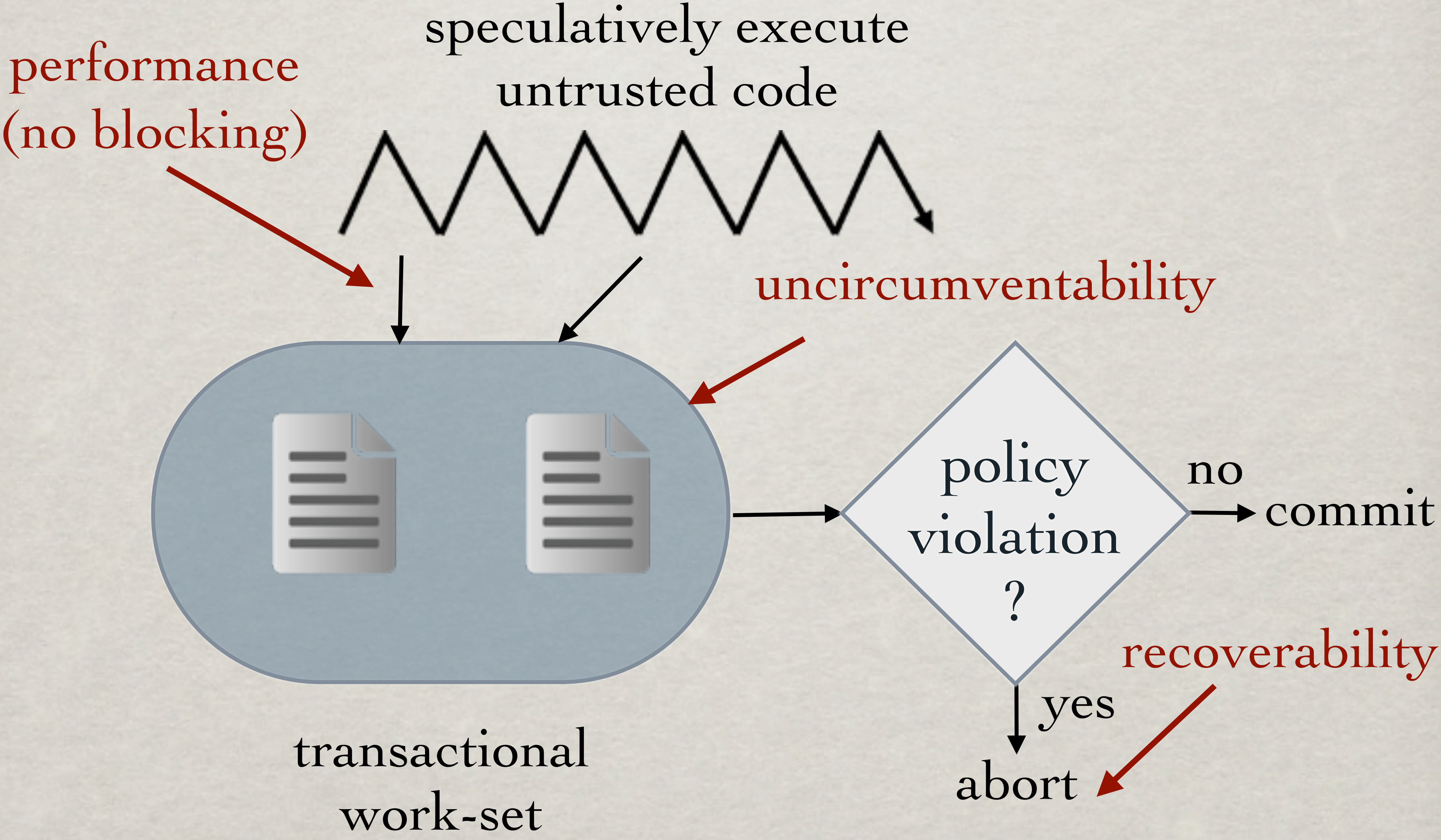
no

commit

yes

abort

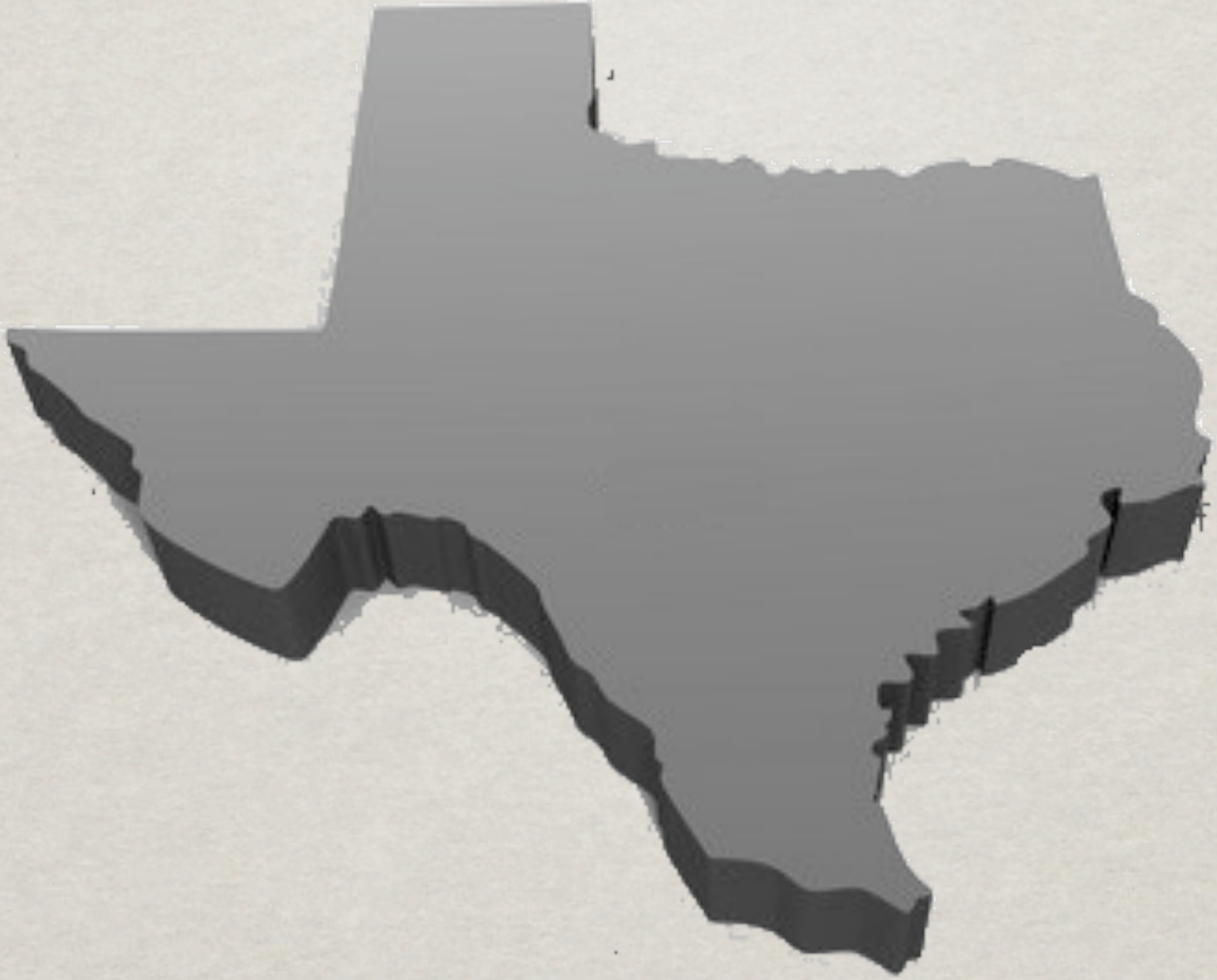
Security needs transactions



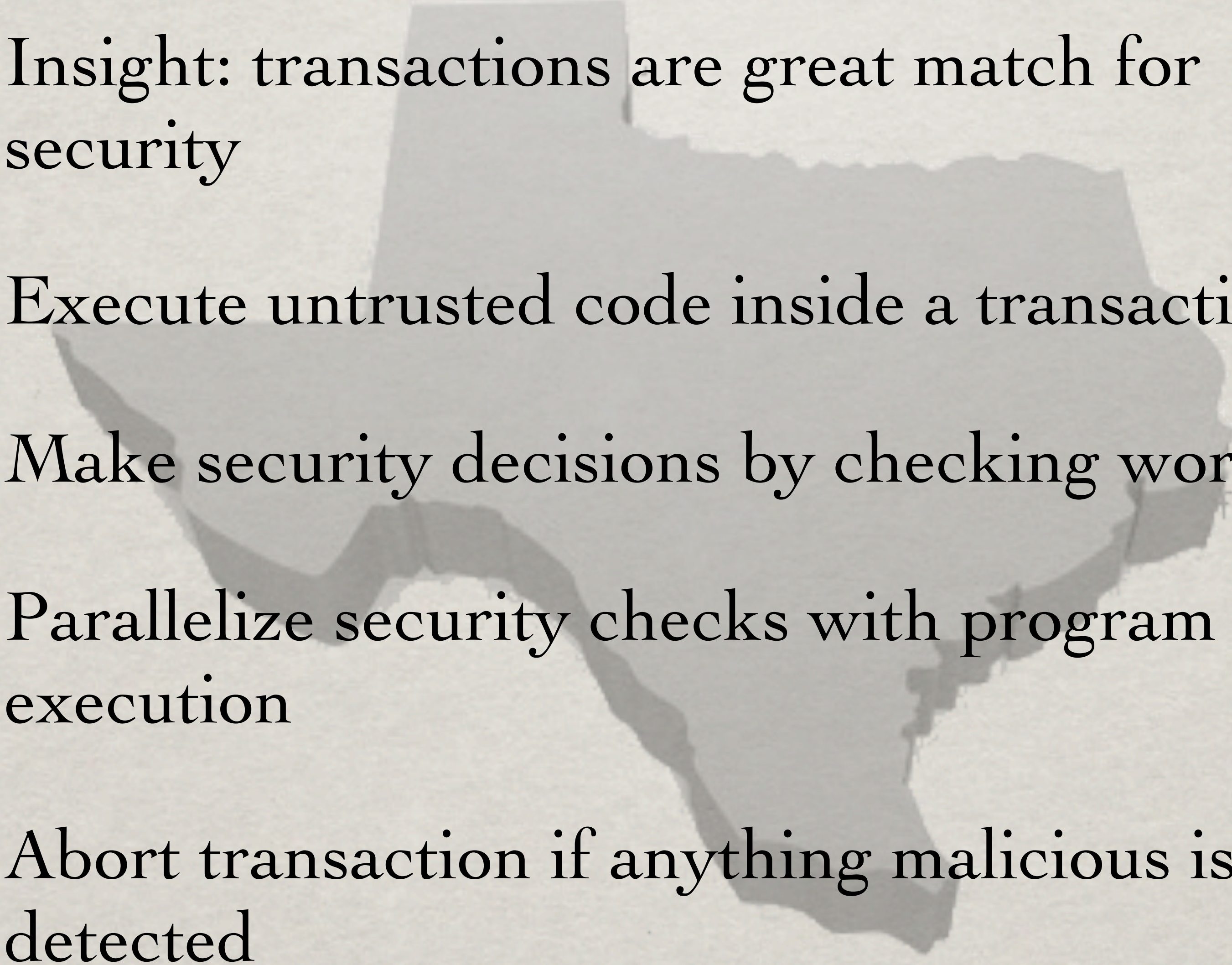
OS support for transactions

- TxOS : Porter et al. SOSP 2009
- Speculative execution support for 150+ system calls
- Provides ACID semantics
- Originally done for handling concurrency

TxBox



TxBox



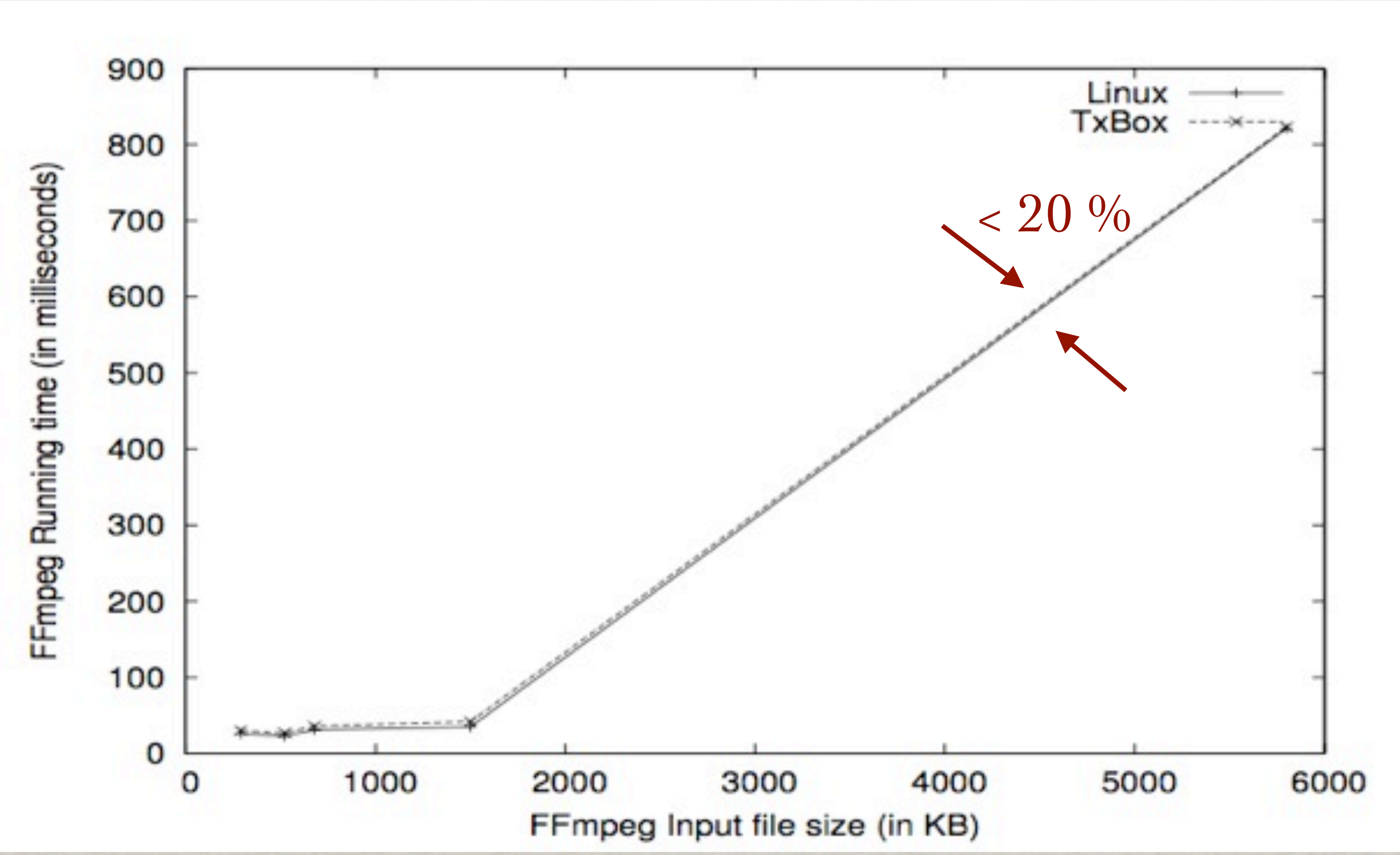
- Insight: transactions are great match for security
- Execute untrusted code inside a transaction
- Make security decisions by checking work-set
- Parallelize security checks with program execution
- Abort transaction if anything malicious is detected

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 ?

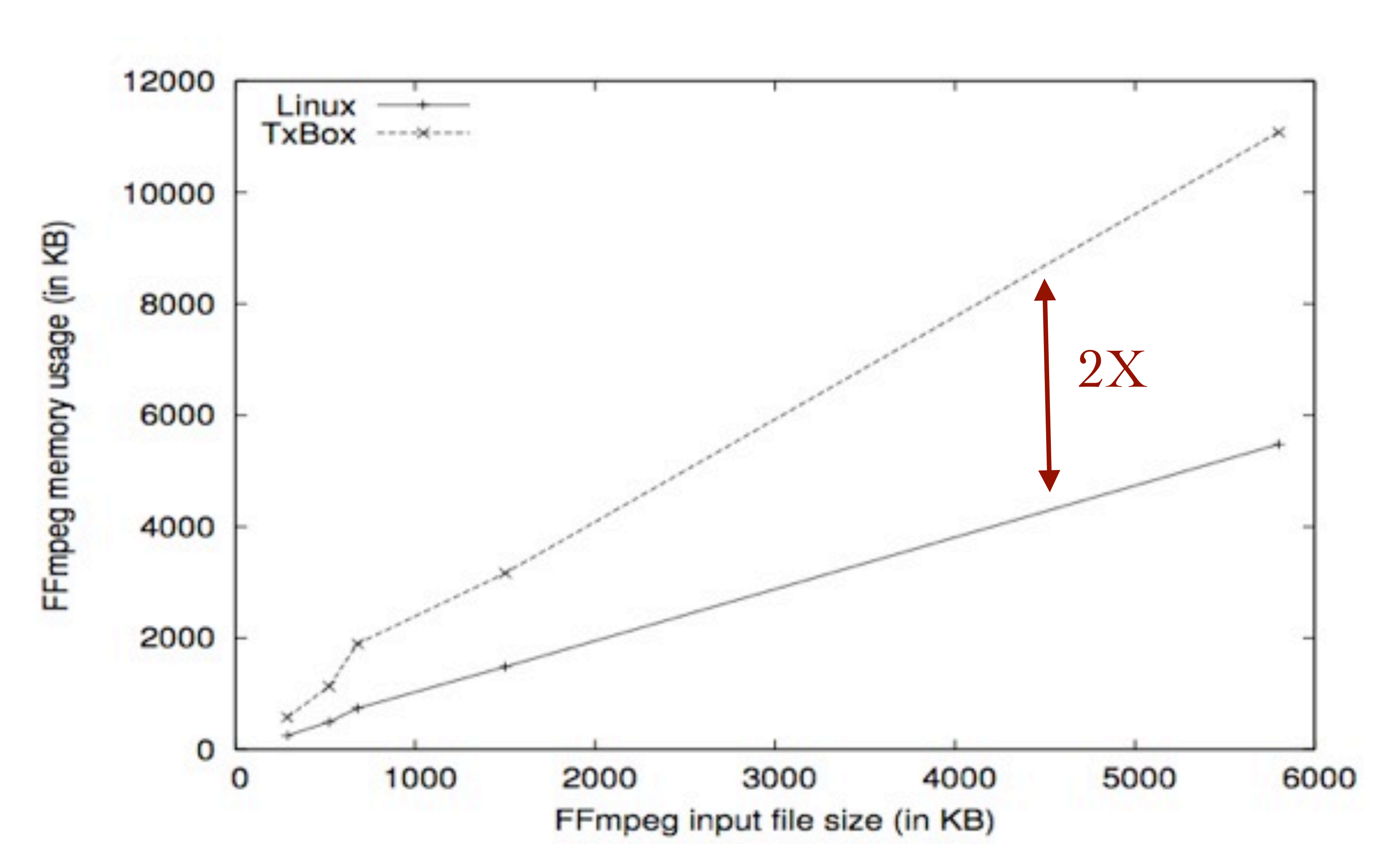
TxBox: performance overhead

- On average TxBox causes less than < 20% runtime overhead compared to Linux

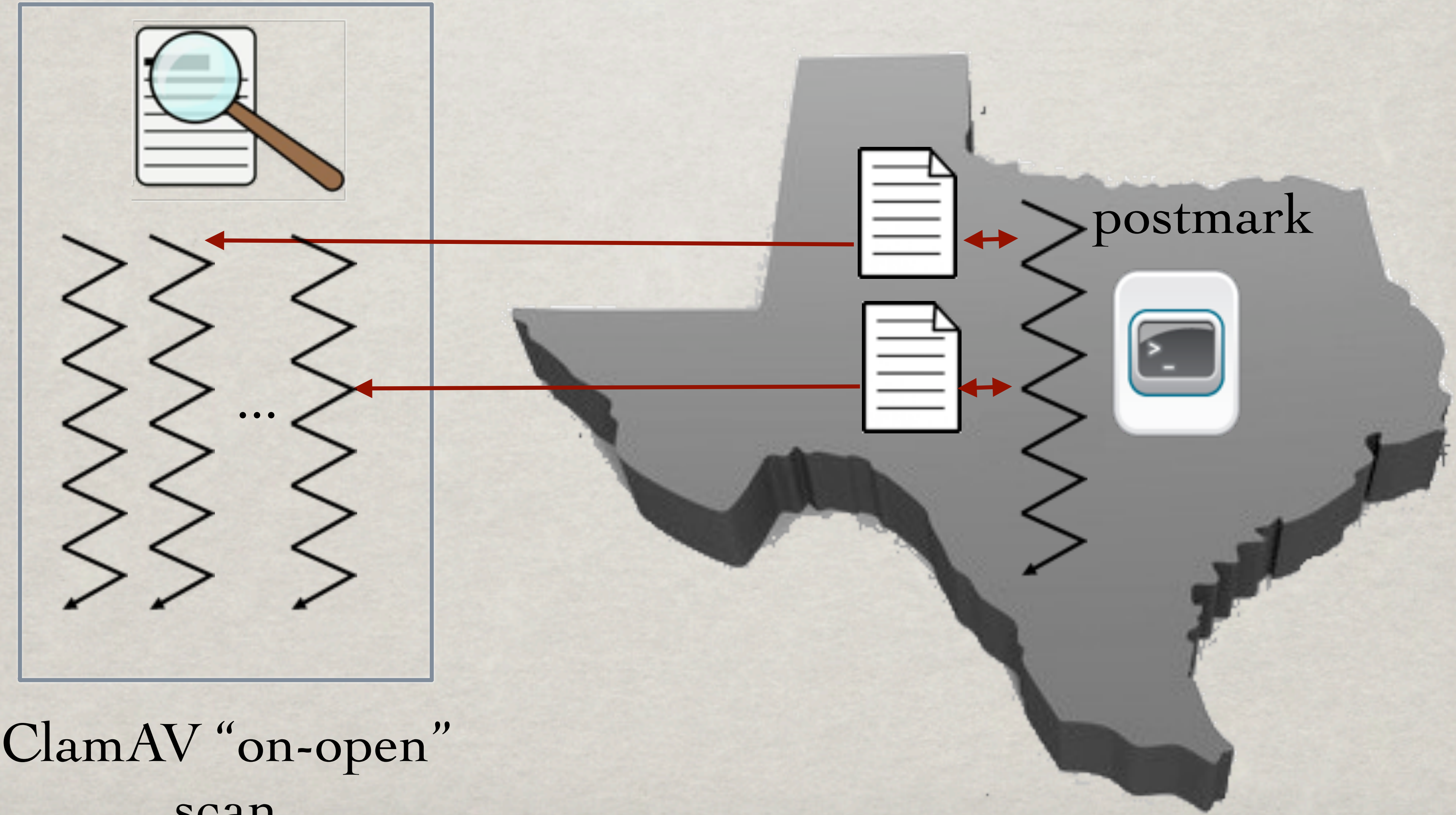


TxBox: memory overhead

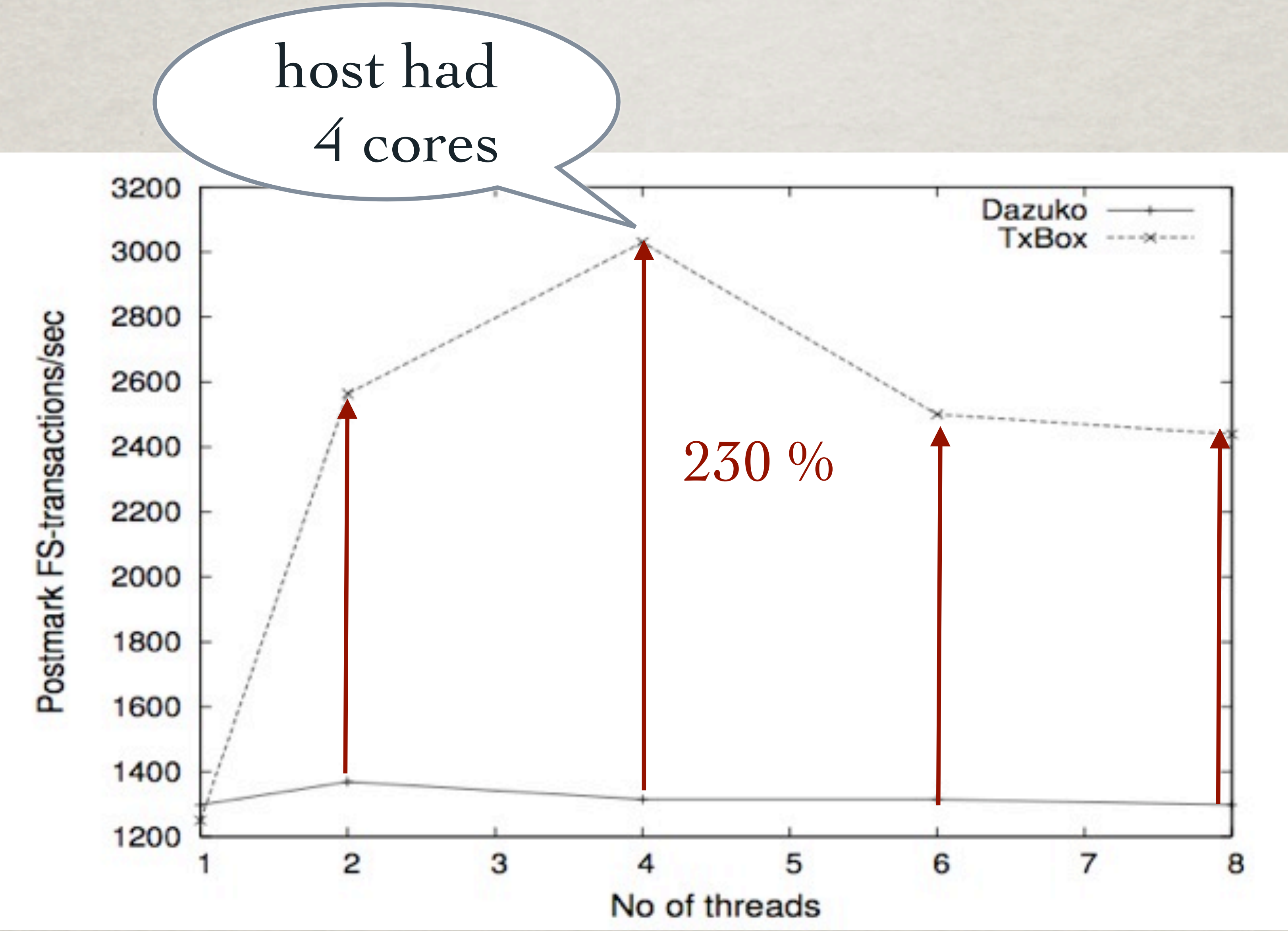
- On average TxBox execution of a process takes 2x more memory compared to regular Linux execution



TxBox: parallel antivirus scanning



TxBox parallelization gain (ClamAV scanning)



host had 4 cores

230 %

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

?

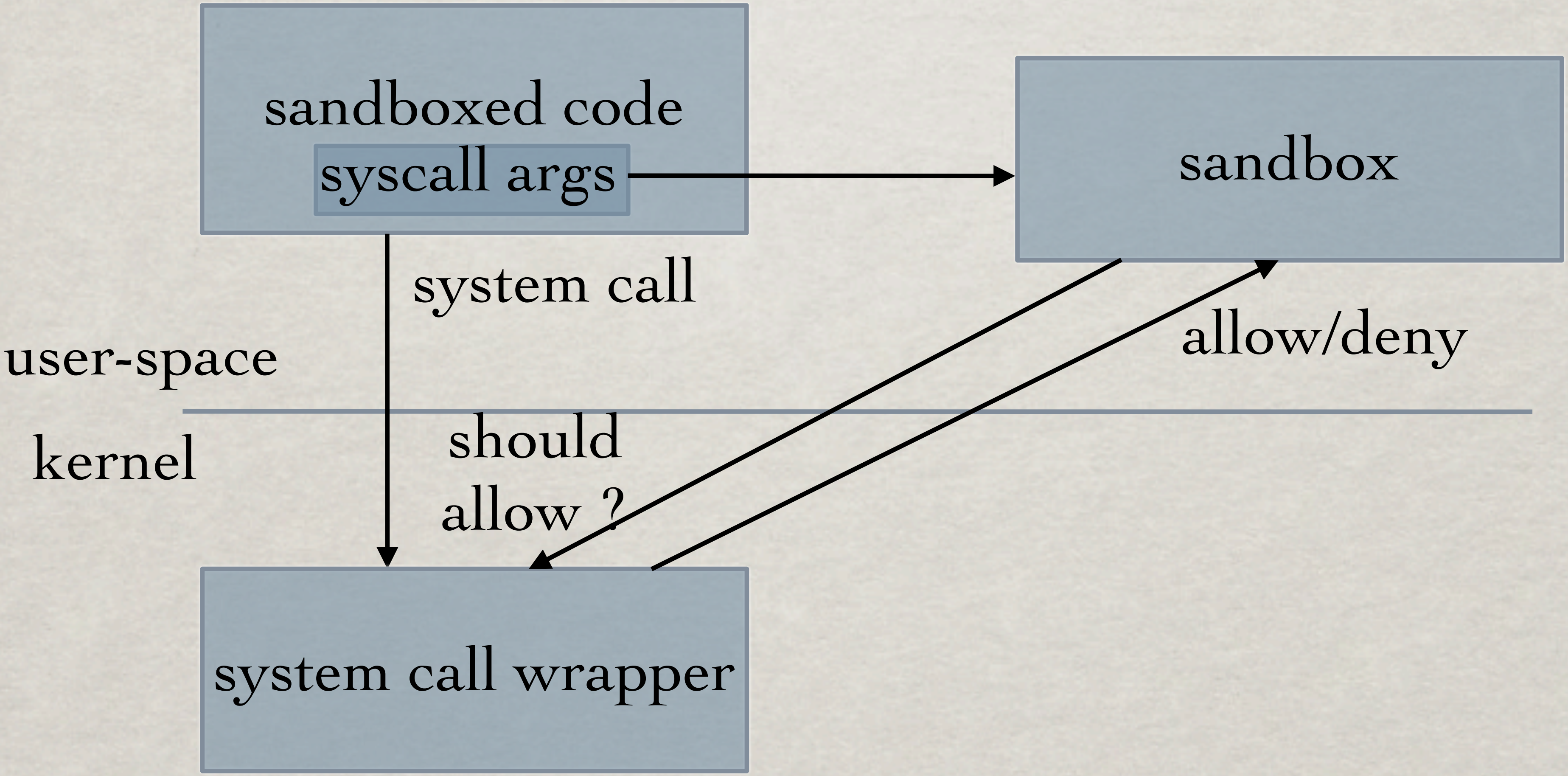
Recoverability: output commit problem

- 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

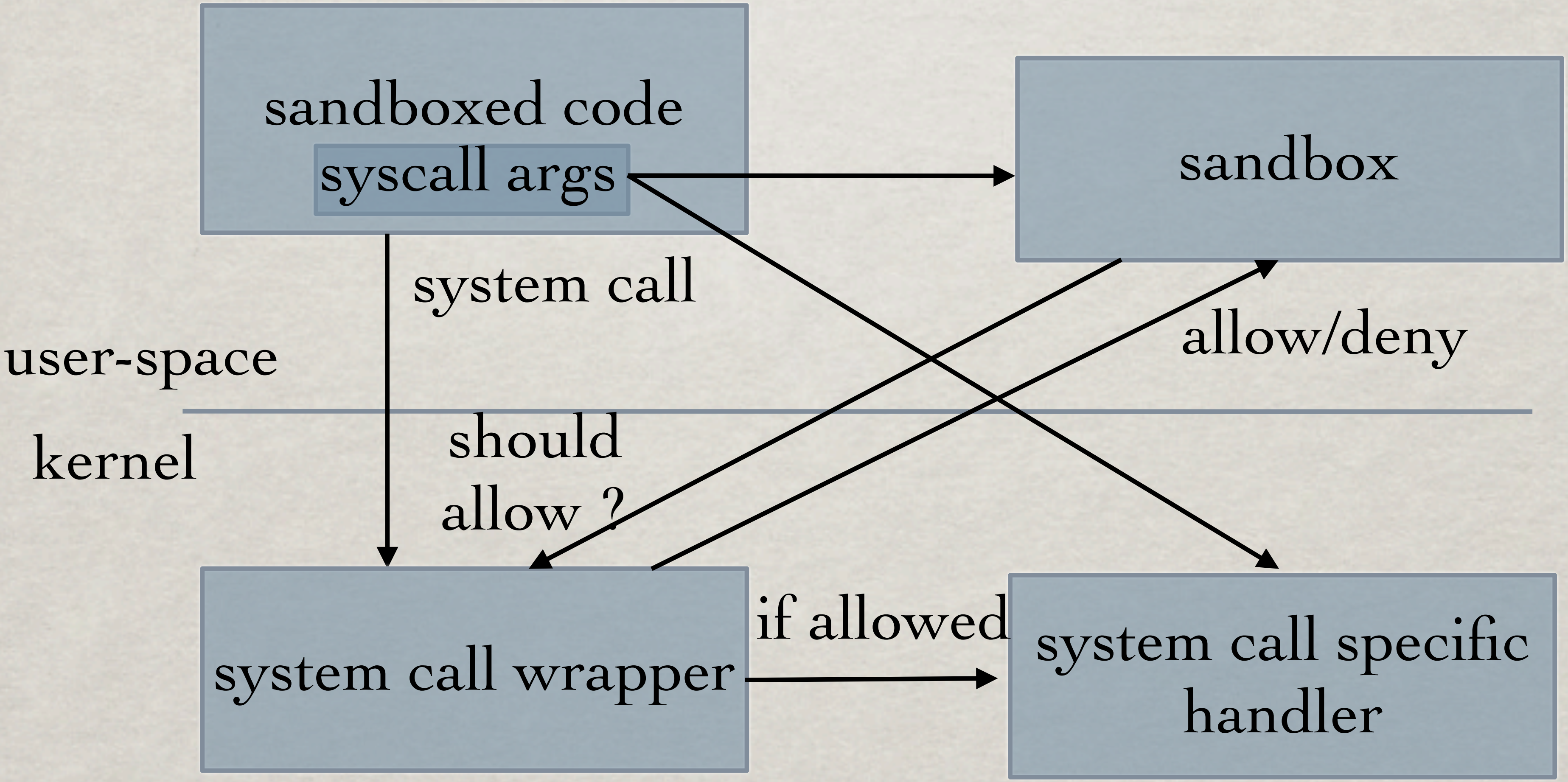
TxBox: implementation issues

- TxOS transactions need cooperative processes calling
  - xbegin
  - xend
- Untrusted processes are not co-operative
  - Support “forced” transactions
- Implement policy manager and policy enforcer
- See paper for details

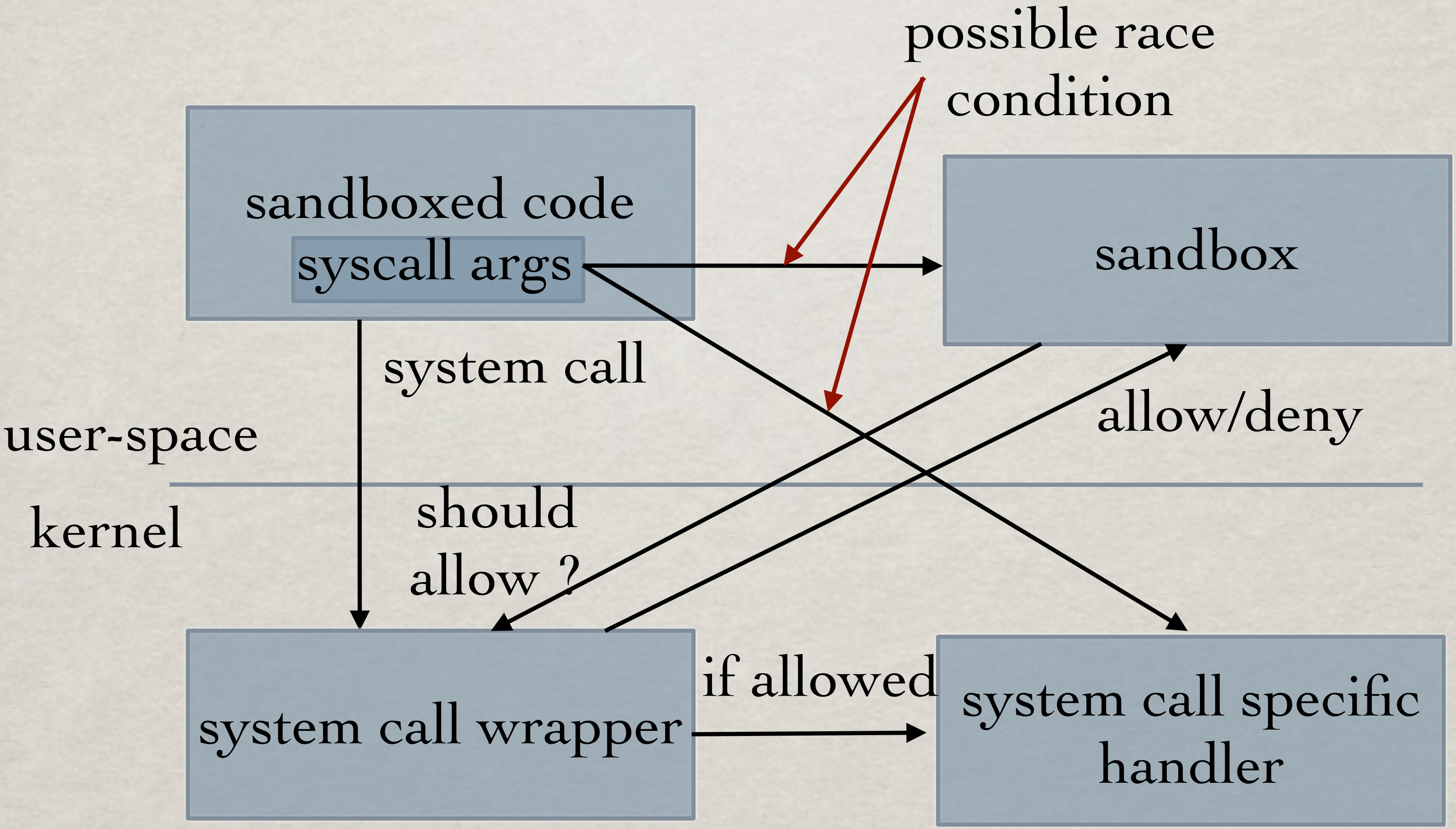
Building sandboxes with system call interposition



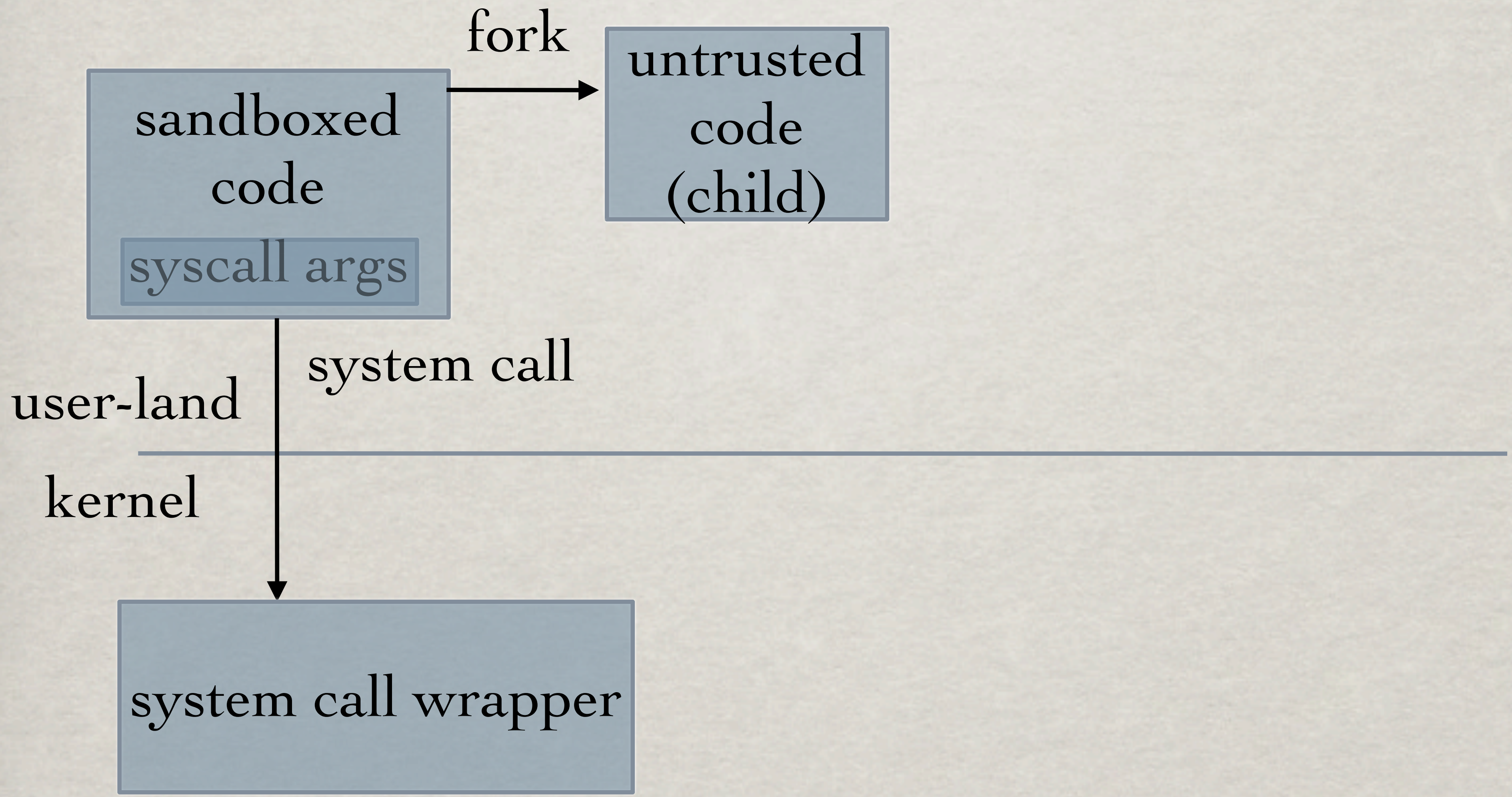
Building sandboxes with system call interposition



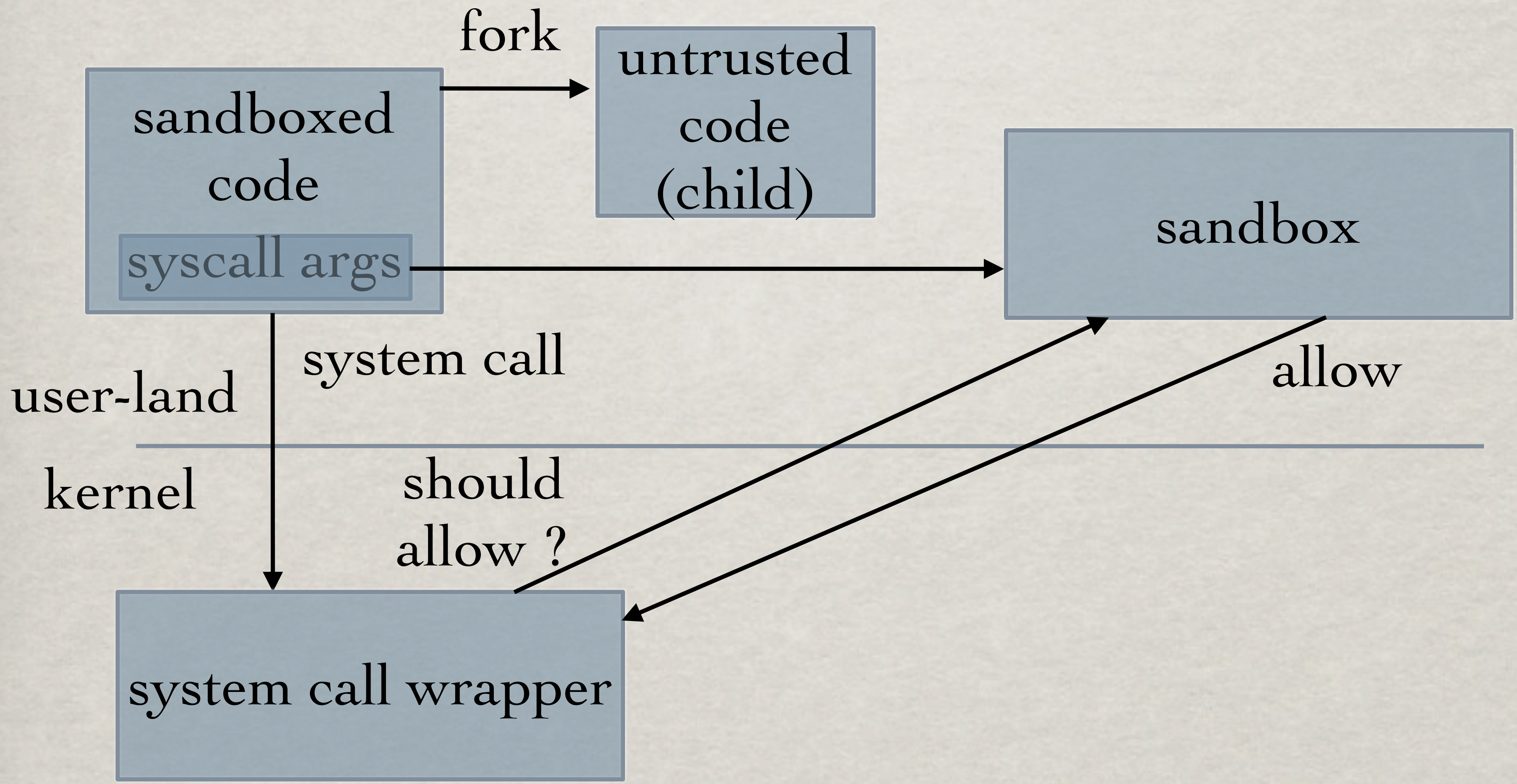
Building sandboxes with system call interposition



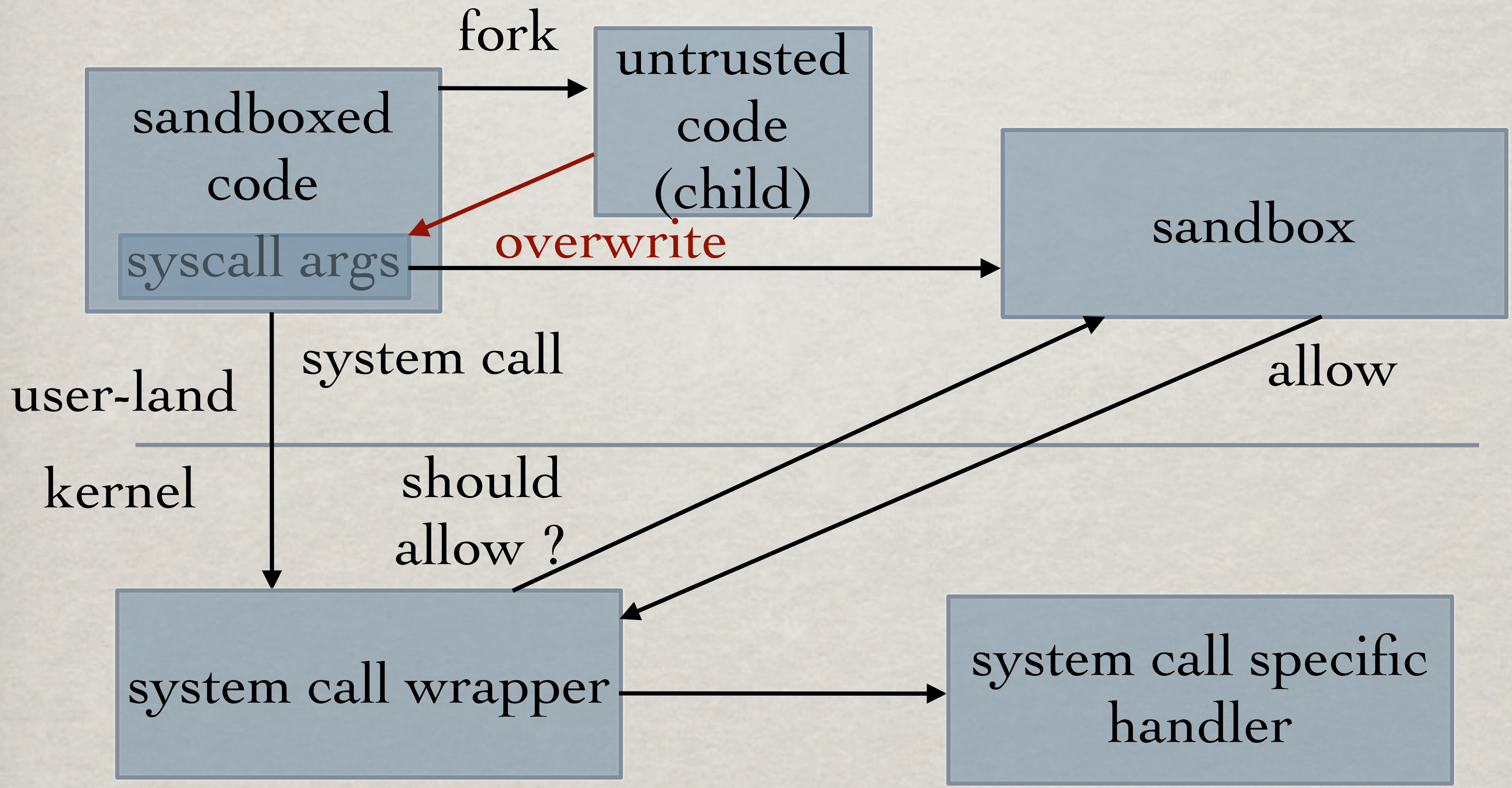
Time of check to time of use (TOCTOU) attacks



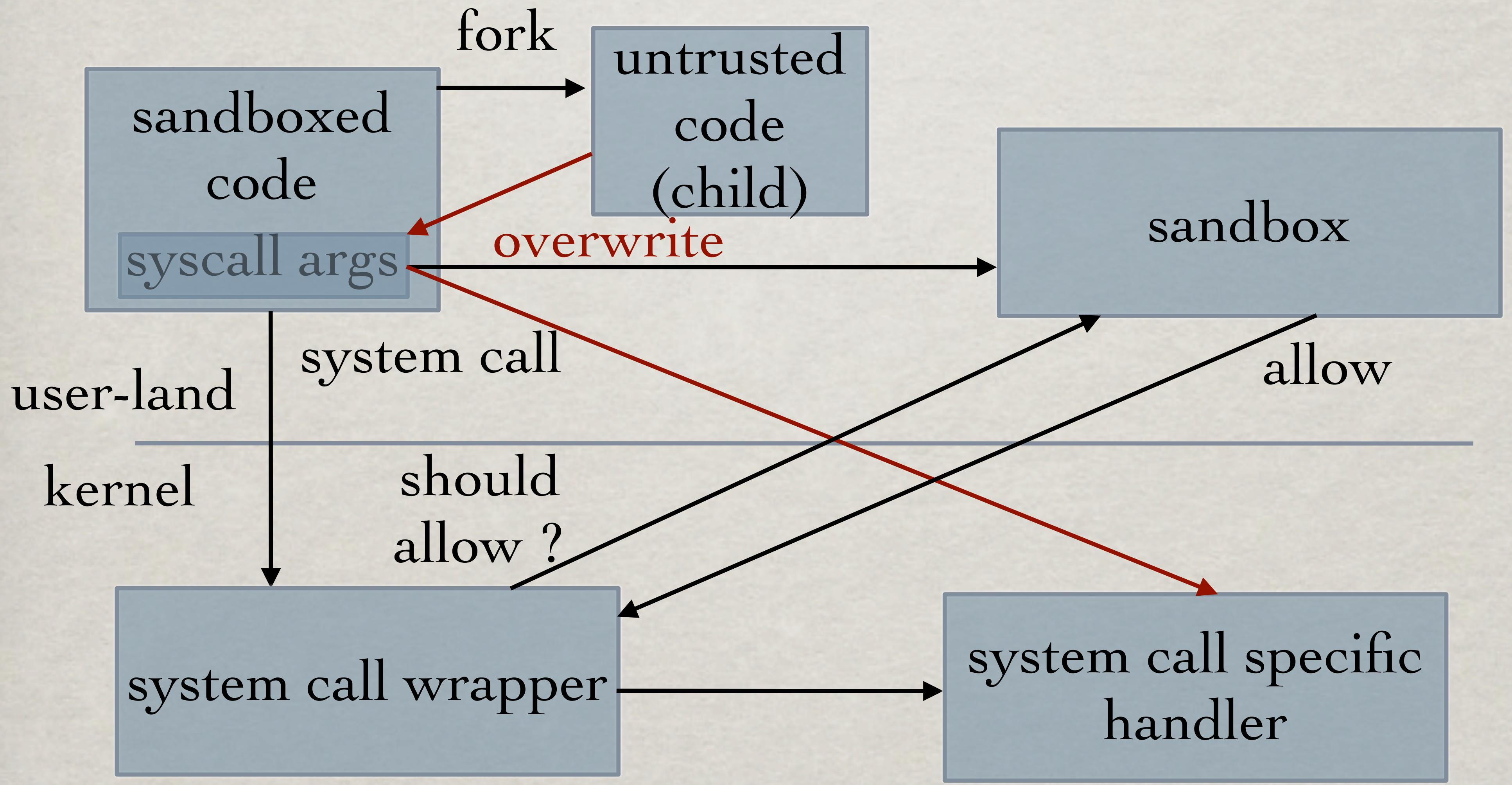
Time of check to time of use (TOCTOU) attacks



Time of check to time of use (TOCTOU) attacks



Time of check to time of use (TOCTOU) attacks


fork
untrusted code (child)
sandboxed code
syscall args
overwrite
sandbox
system call
allow
user-land
kernel
should allow ?
system call wrapper
system call specific handler