Thoughts on Client Systems Security

Joanna Rutkowska
Invisible Things Lab

SSTIC 2011, Rennes, France, June 2011

Why client systems security is important?

If your client device (laptop, tablet, phone) is compromised...

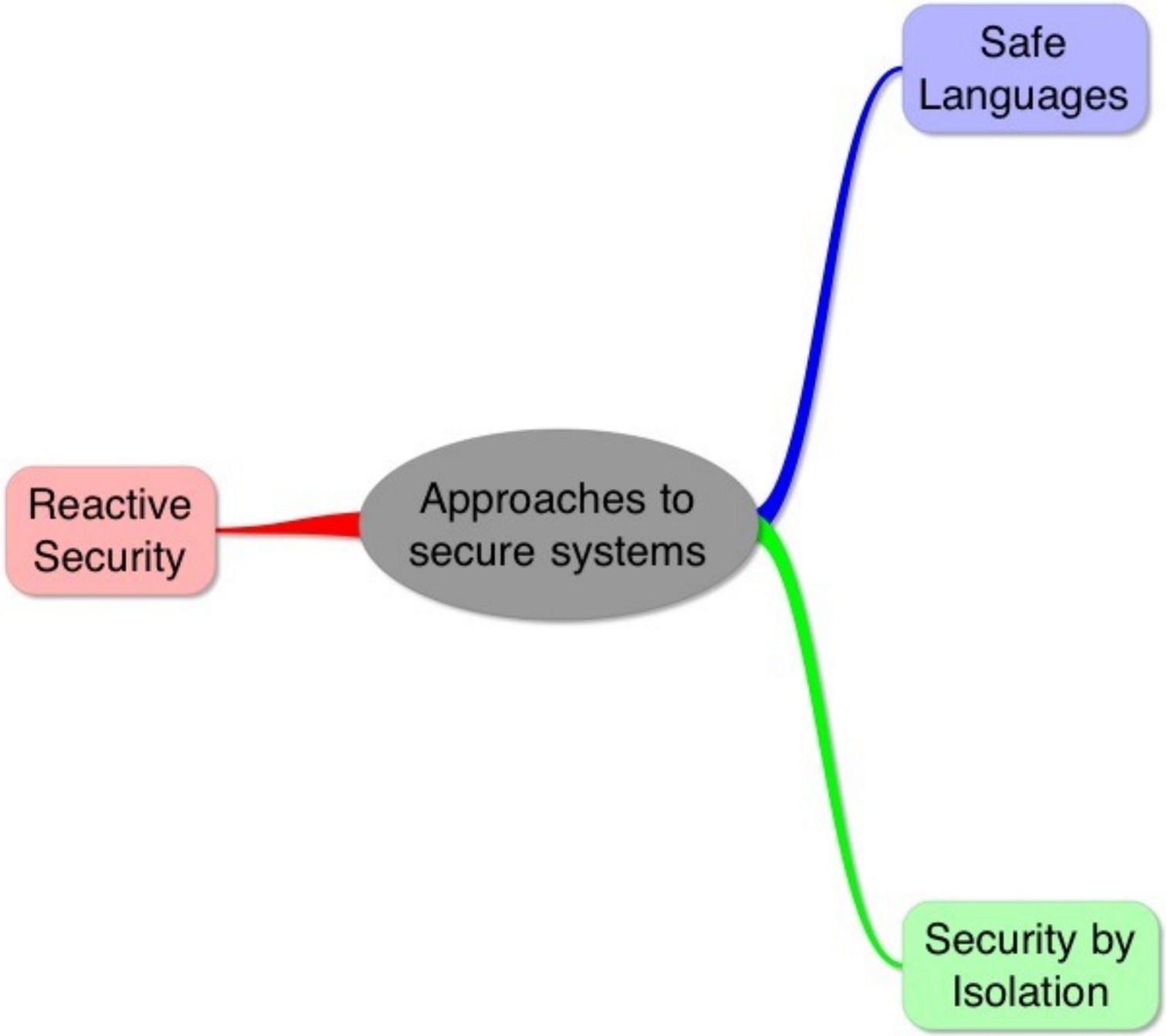
... all the security is lost!

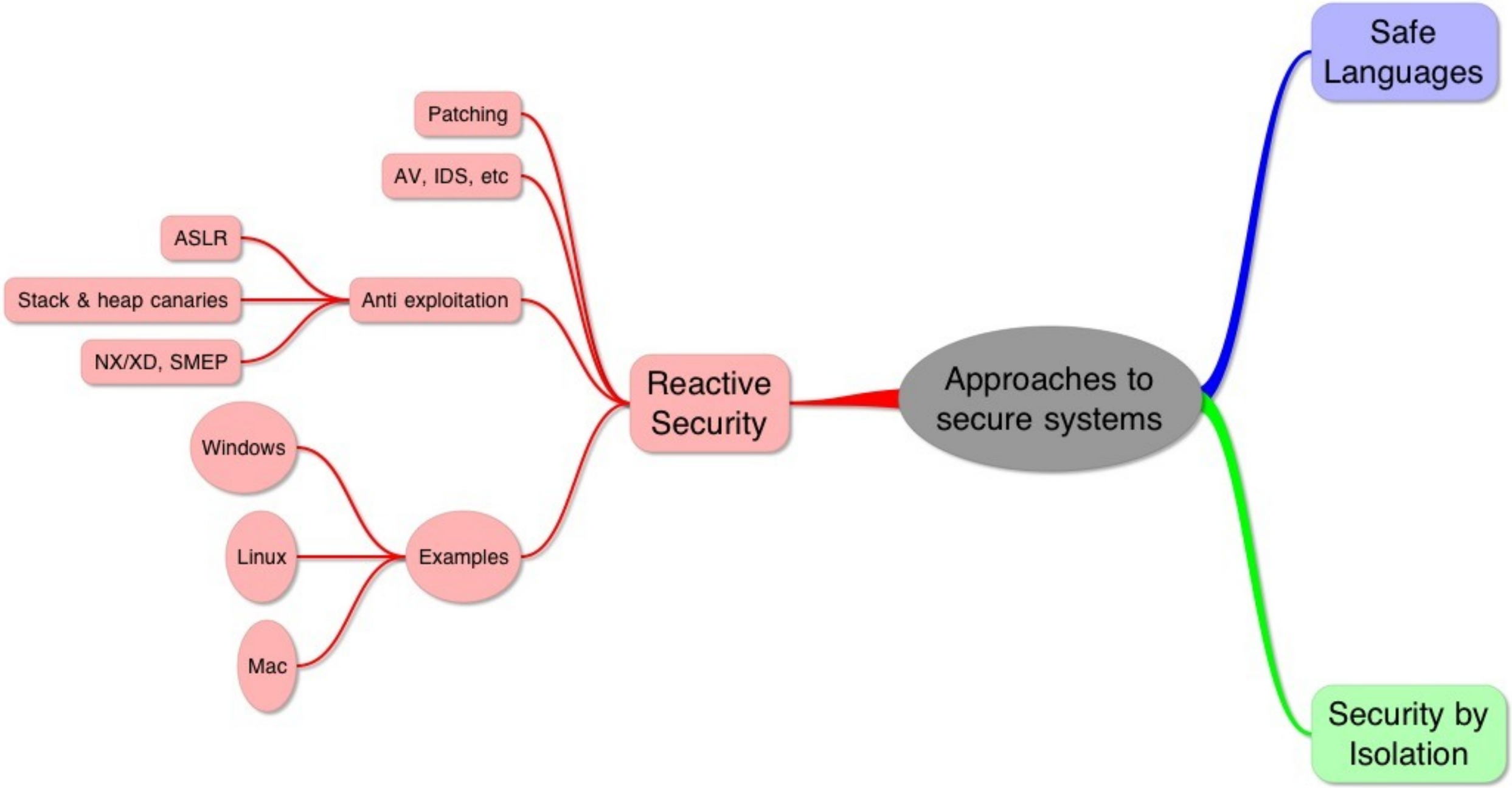
Client systems are your **eyes** and **fingertips**

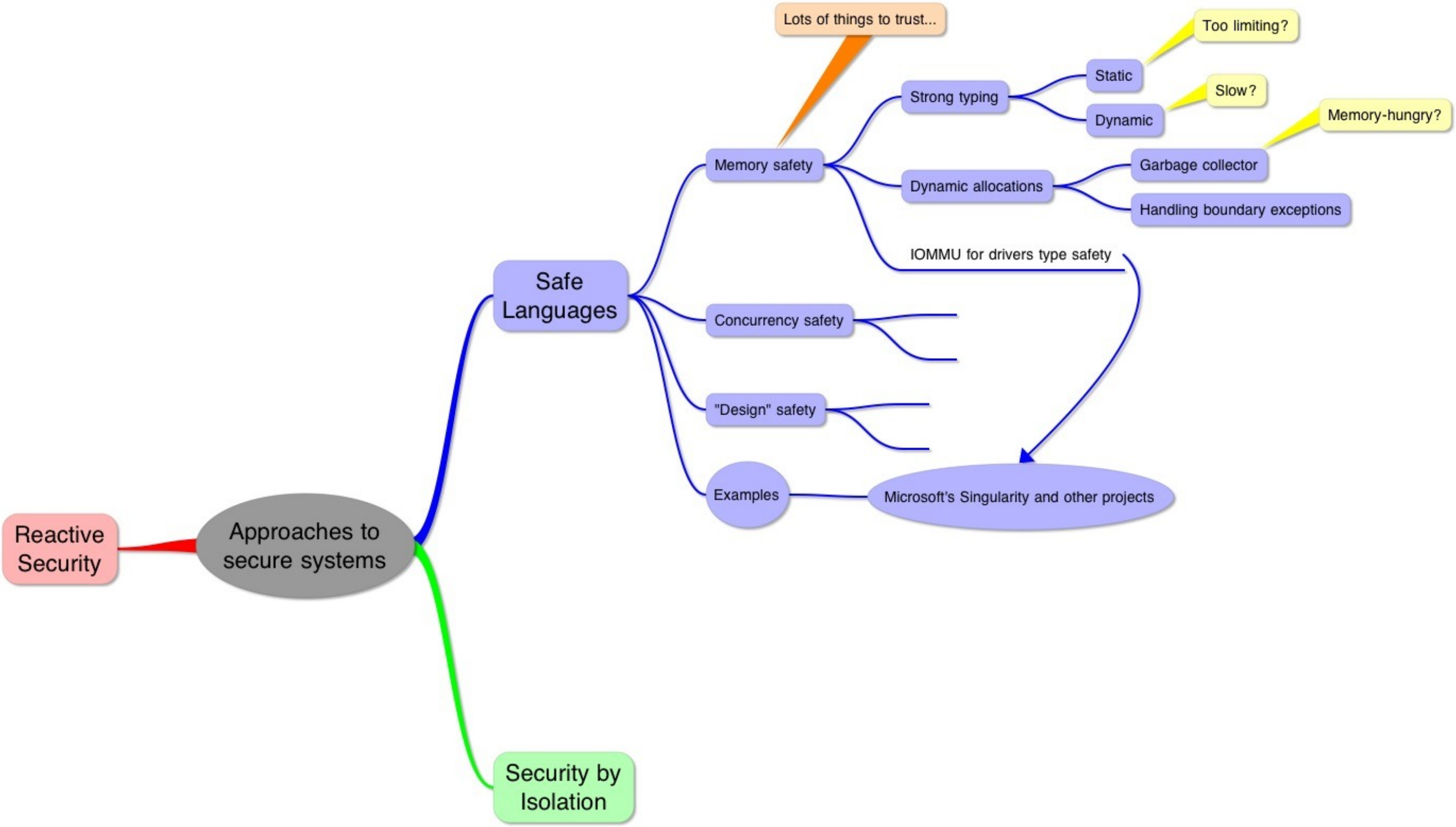
The client OS can see what you see on the screen (decrypted)

The client OS can pretend to be you

Approaches to building secure (client) systems


Safe Languages
Reactive Security
Approaches to secure systems
Security by Isolation


Safe Languages
Patching
AV, IDS, etc
ASLR
Stack & heap canaries
Anti exploitation
NX/XD, SMEP
Reactive Security
Approaches to secure systems
Windows
Linux
Examples
Mac
Security by Isolation


Safe Languages
Approaches to secure systems
Reactive Security
Security by Isolation
Memory safety
Lots of things to trust...
Strong typing
Static
Too limiting?
Dynamic
Slow?
Dynamic allocations
Garbage collector
Memory-hungry?
Handling boundary exceptions
IOMMU for drivers type safety
Concurrency safety
"Design" safety
Examples
Microsoft's Singularity and other projects

- Approaches to secure systems
  - Reactive Security
  - Safe Languages
  - Security by Isolation
    - Small TCB!
      - Thin microkernels & hypervisors
        - Formal verification
        - Common-sense verification
      - **Untrusted subsystems!**
        - Networking
        - Storage
        - Drivers virtualization (IOMMU)
        - GUI?
    - Examples
      - Xen Client
      - Qubes
      - Google Chrome

Security by Isolation: Goals

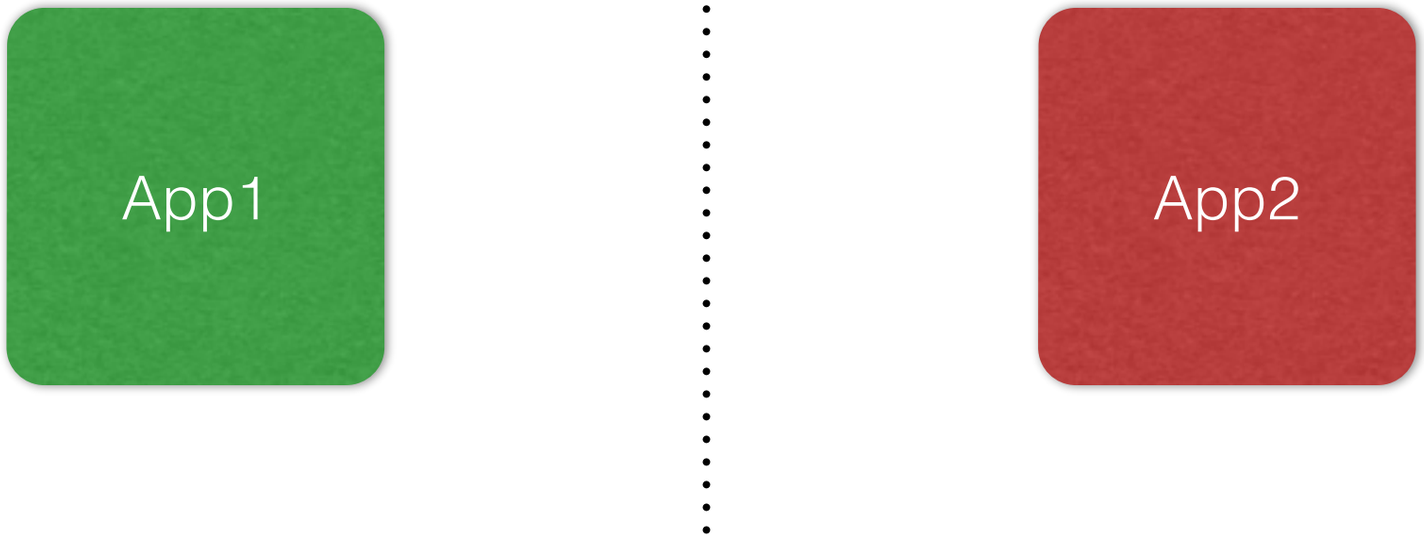
Isolation between two apps...

App1

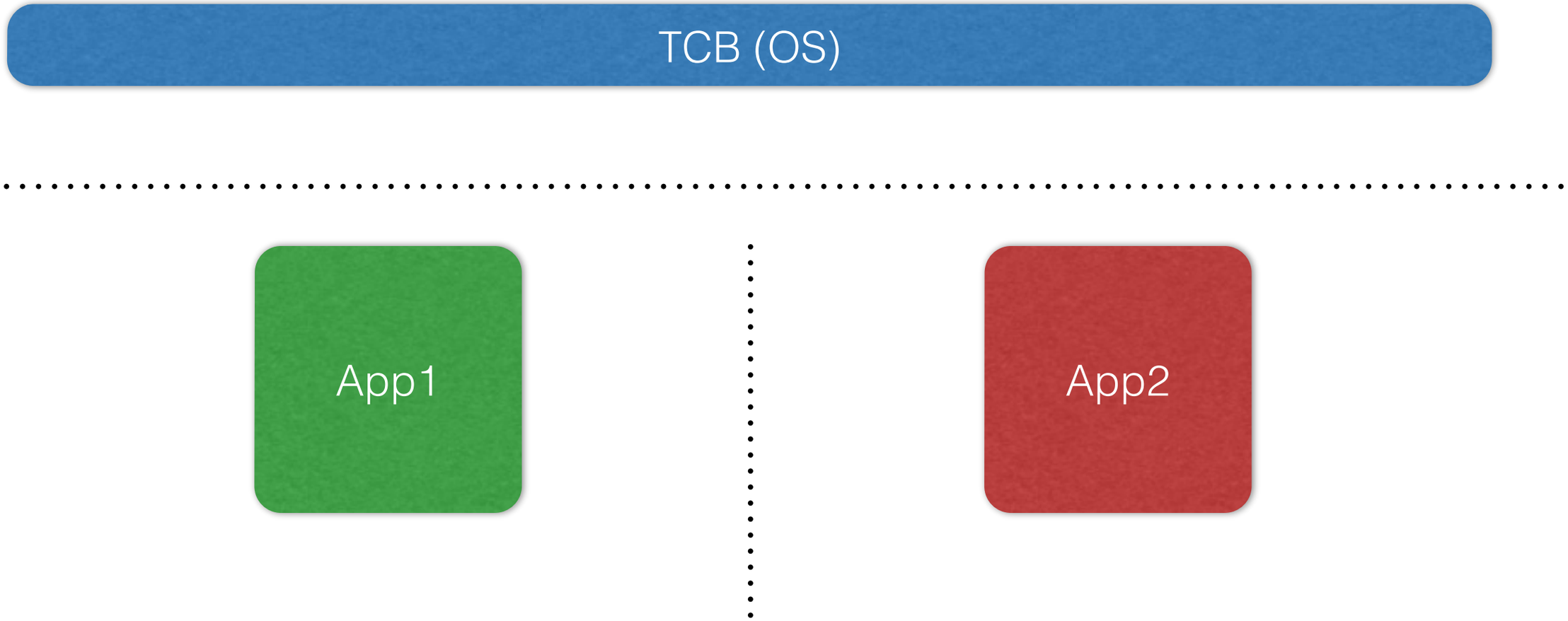
?

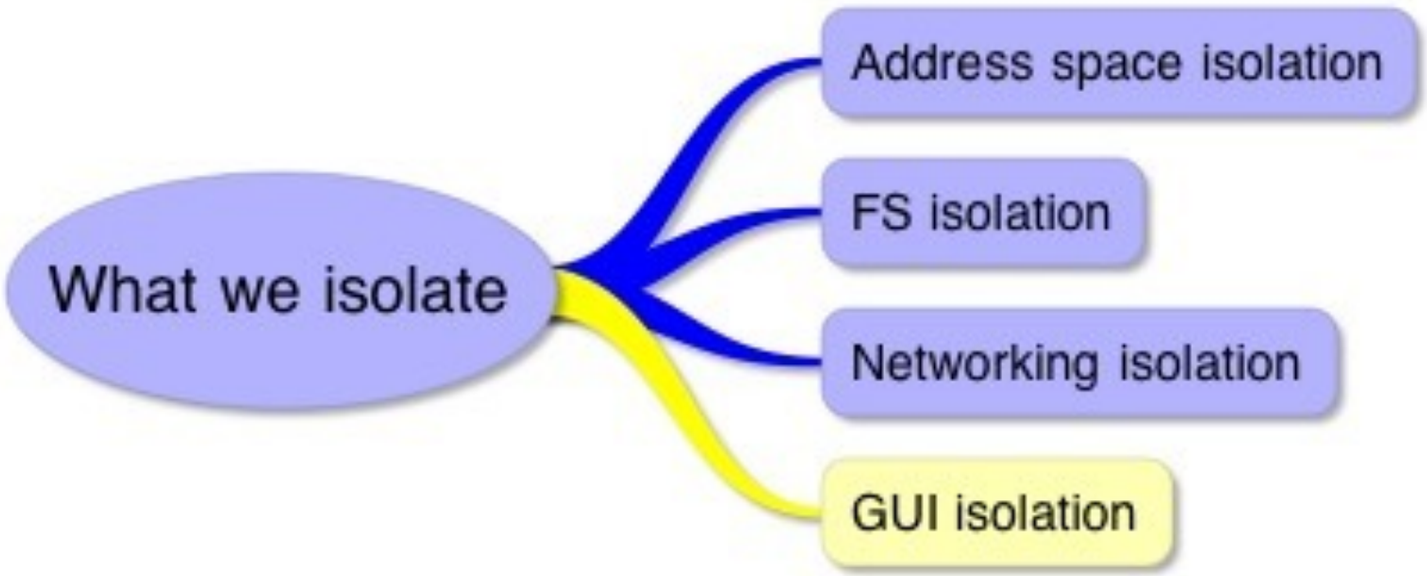
App2

Isolation between two apps...



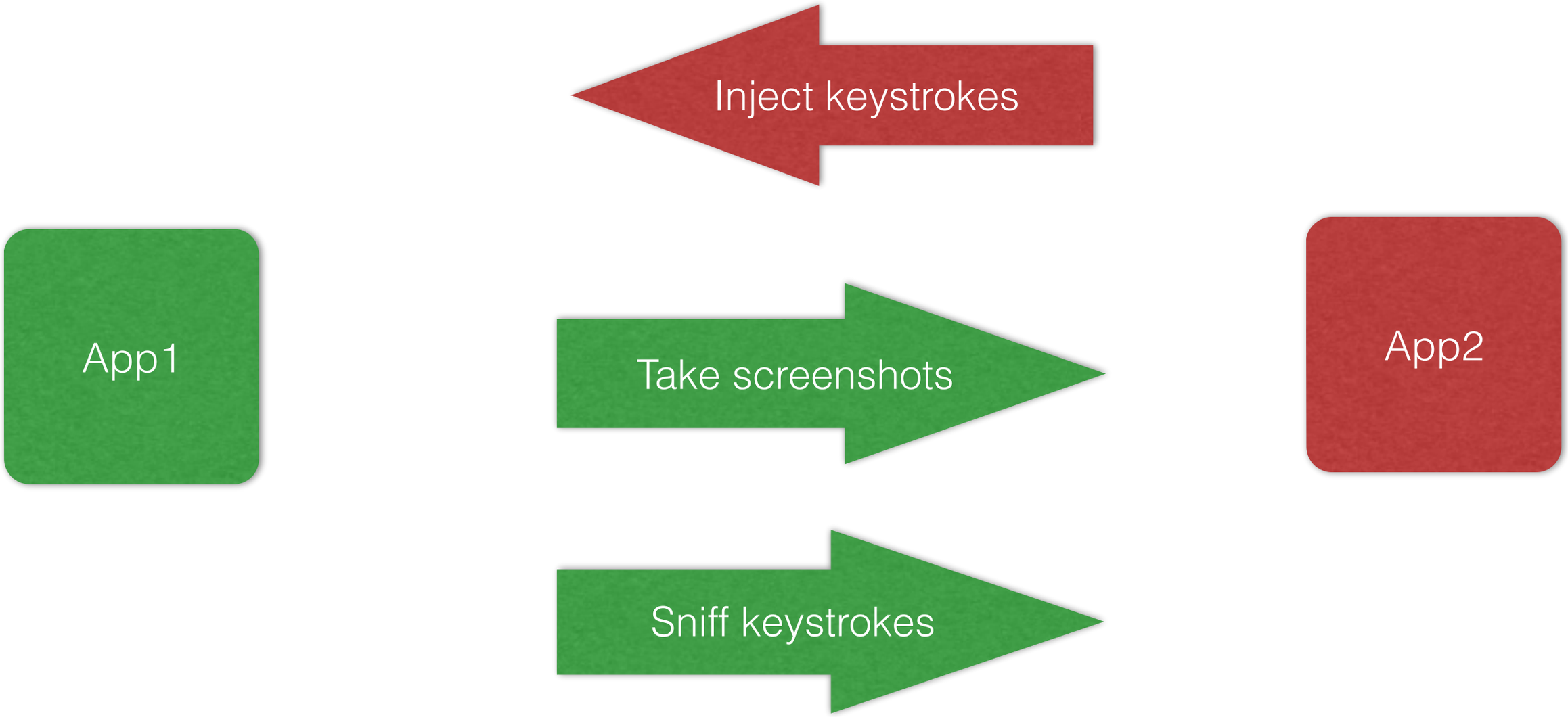
Isolation between two apps and the OS...



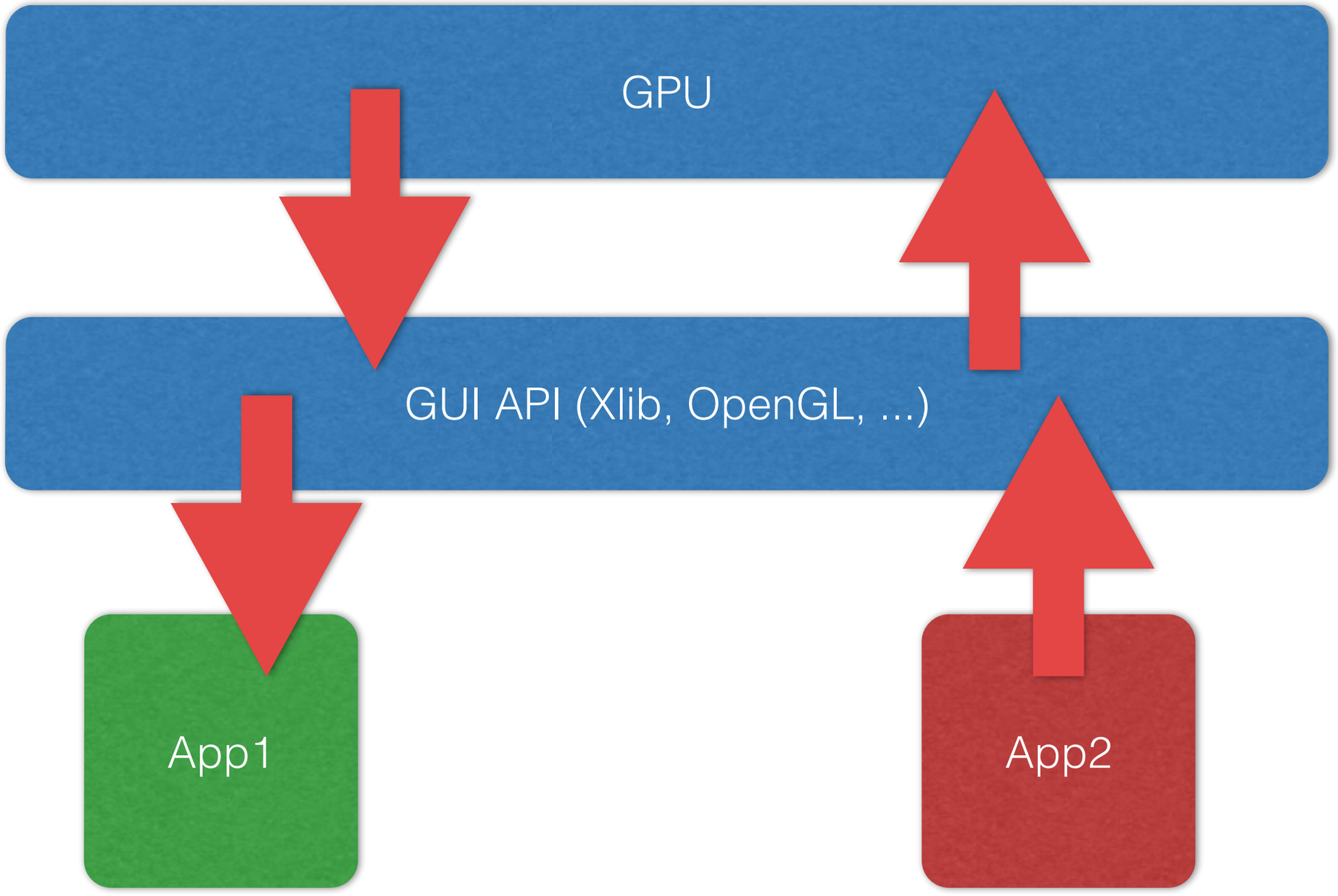

What we isolate
Address space isolation
FS isolation
Networking isolation
GUI isolation

GUI-level isolation

Lack of GUI isolation on many Windowing Systems...


Inject keystrokes
App1
Take screenshots
App2
Sniff keystrokes

Fat GUI APIs that are likely to be buggy (and exploitable)


GPU
GUI API (Xlib, OpenGL, ...)
App1
App2

Work email

Tetris

Bank Browser

Personal Browser

We don't want two apps to be able to interact with each other via X/OpenGL/GPU!

(Xorg people still don't get it, after 20+ years...)

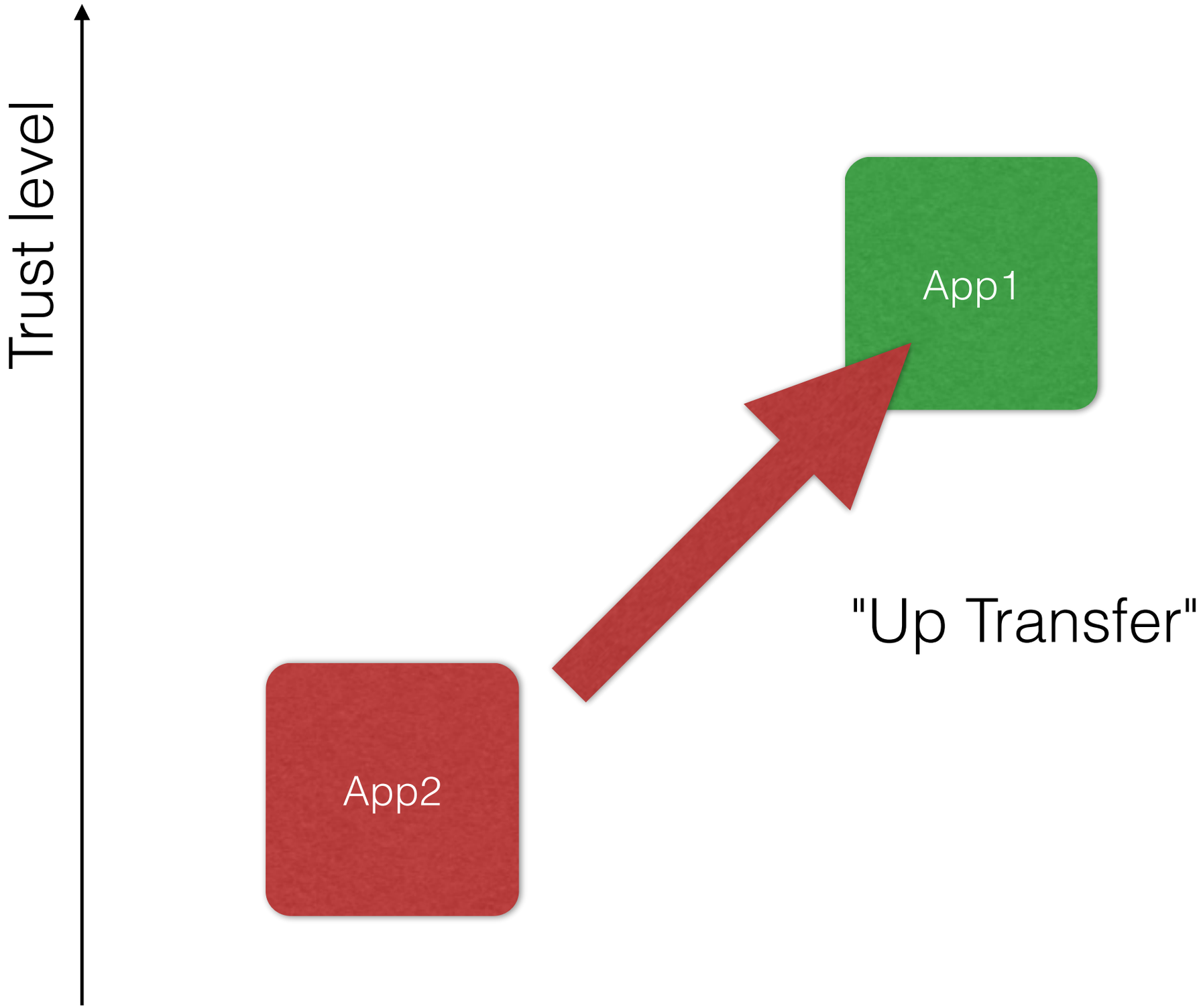
Anyway...

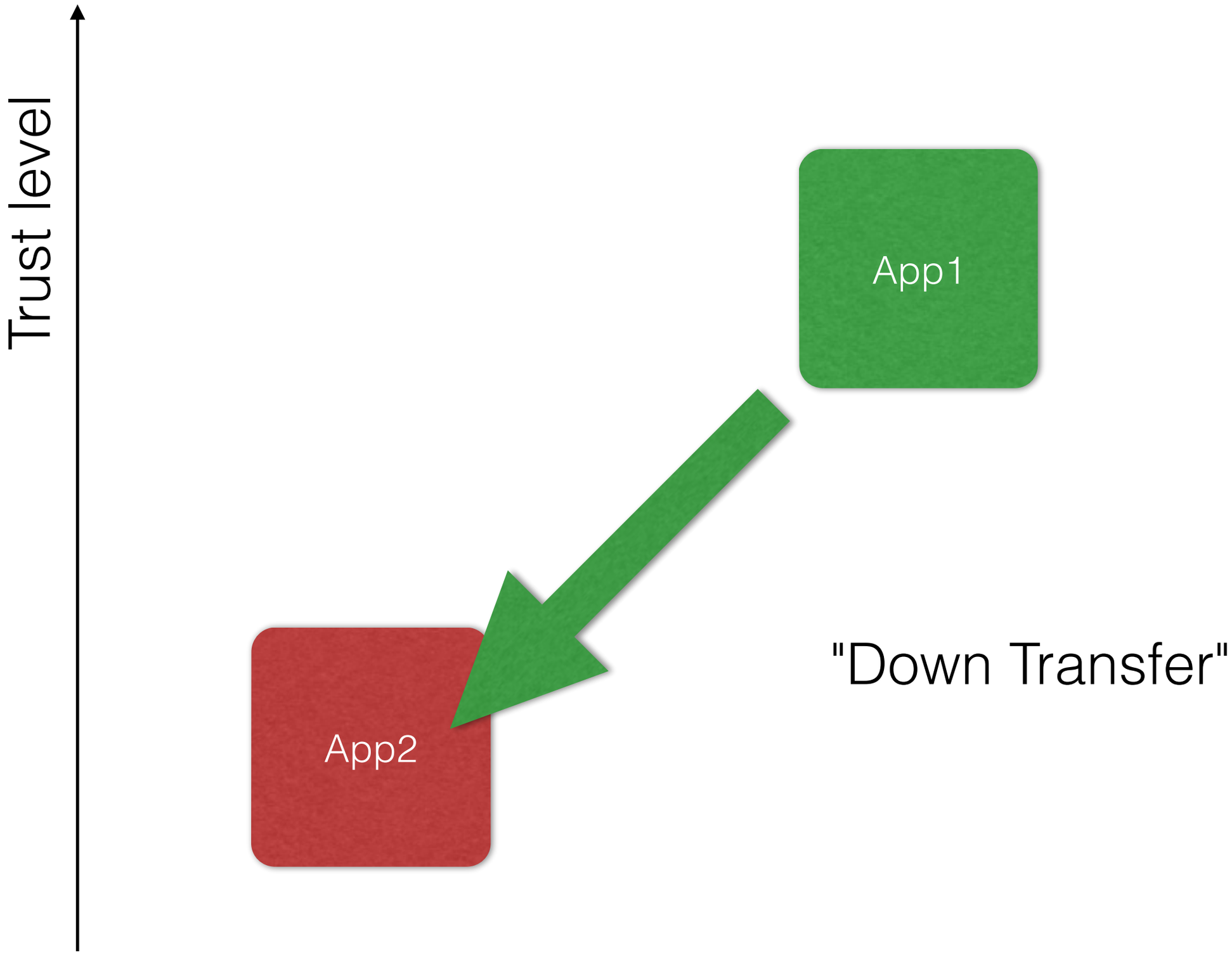
Let's imagine we implemented strong isolation...

We still must allow the user to bypass it sometimes!

Data flows between domains

Down-transfers vs. Up-transfers


Trust level
App1
App2
"Up Transfer"



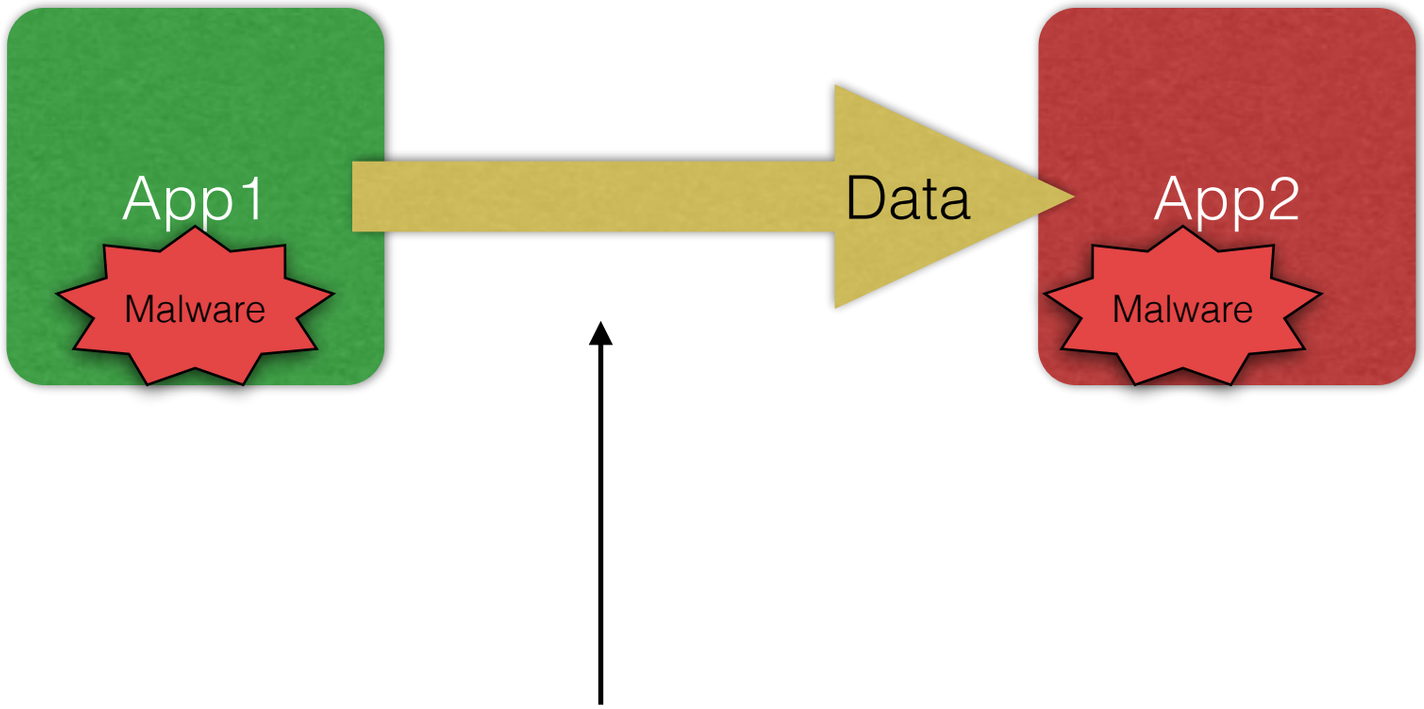
"Down Transfer"

"Traditional" school of thought:

Never allow **down-transfers!**

Even between two *cooperating* domains!

Rationale: never allow to move more sensitive data (e.g. Embassy cables) to less trusted domain (e.g. The Internet)



OS should never allow for this flow!

This requires elimination/drastic reduction of all potential **cooperative covert channels** between the apps/domains!

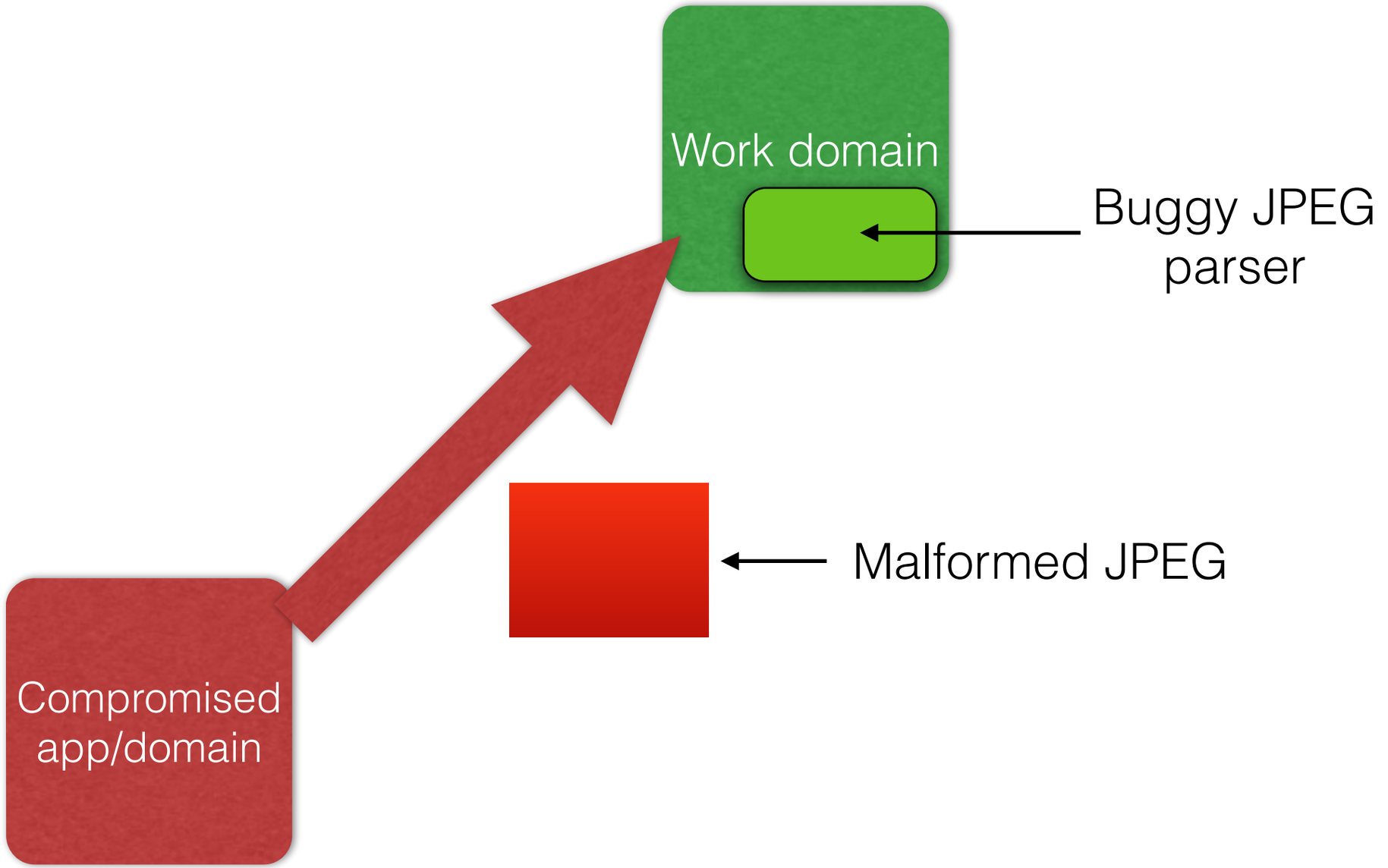
I seriously doubt this is possible on modern x86 hardware...

- Covert channels via CPU cache
- Covert channels via GUI/GPU
- Covert channels via networking
- Covert channels via other subsystems
- ?

"Qubes" school of thought:

Avoid **up-transfers!**

Rationale: an up-transfer can potentially compromise a buggy app in the destination domains (untrusted input processing)



Some up-transfers are difficult to avoid...

Copying a link found on the Internet, and emailing it to a colleague at work

Copying a cool cartoon found on the Internet into work confidential report/presentation

Solution: use trusted converters, e.g. for all JPEGs?

- Trusted converters
  Written in safe languages
  - Perhaps we can ensure _the converter_ won't be exploited...
    - Memory safety
    - Concurrency safety
    - ...
  - ...but how to ensure the converted file will not exploit the dest app?
    - What if the bug in the dest app can be triggered by _legal_ file format?
  - Need to handle each file type separately
    - JPEG
    - PDF
    - ...
  - How to handle clipboard?
    - It's just a string of bytes...
    - Might be pasted into many different apps...
    - Each app might interpret it differently
    - E.g. An URL string

Another types of problems related to file sharing is

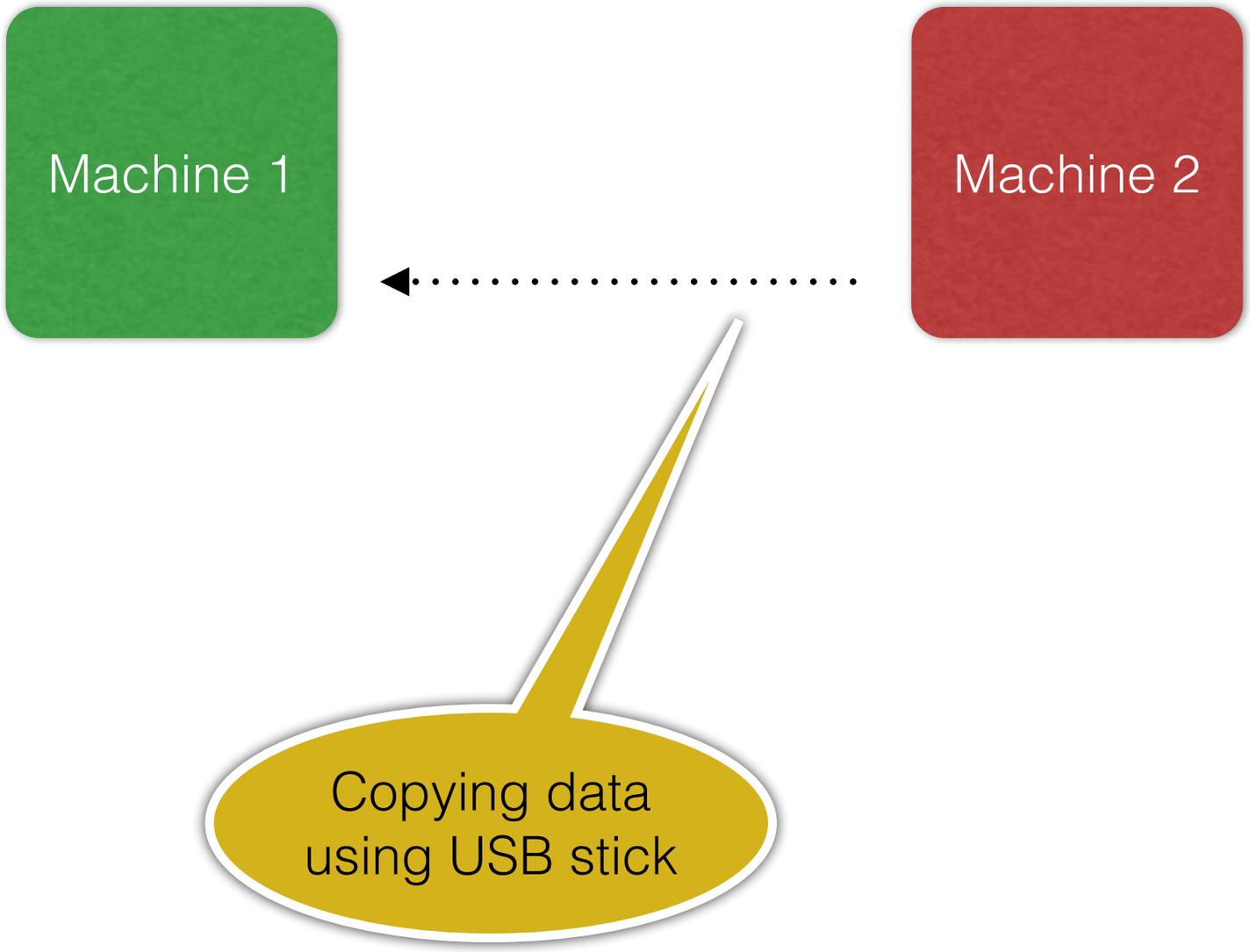
FS Metadata parsing

Two air-gapped systems

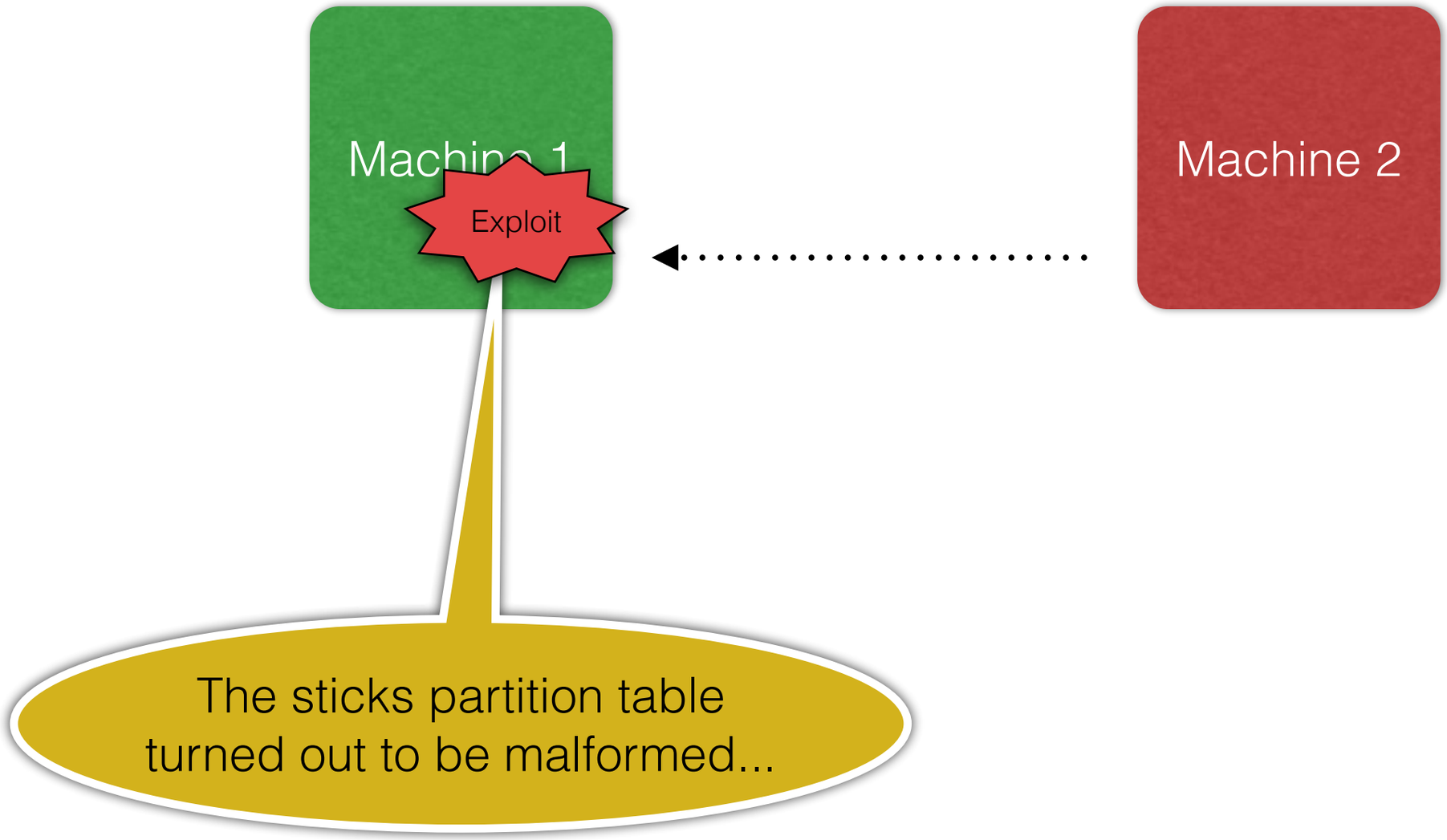
Machine 1

Machine 2

Two air-gapped systems



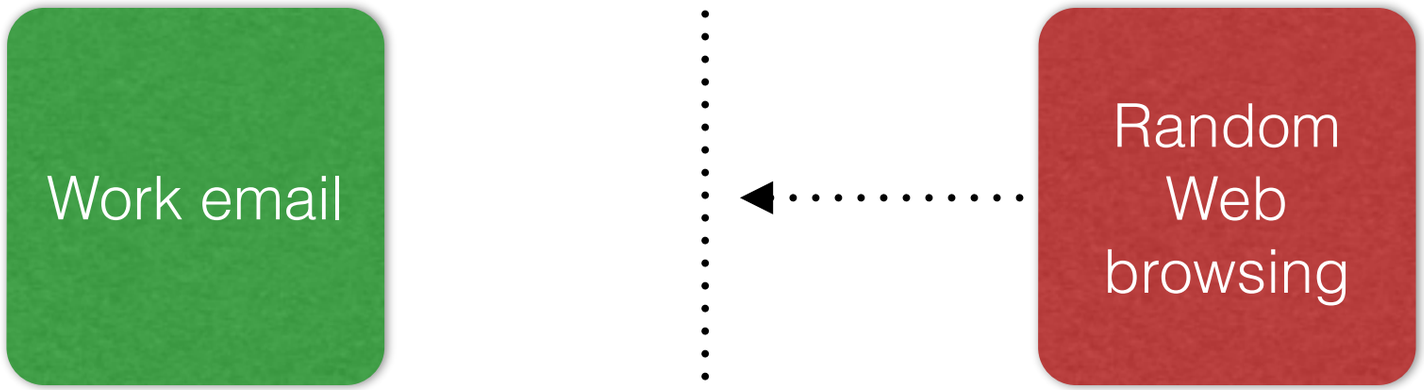
Two air-gapped systems

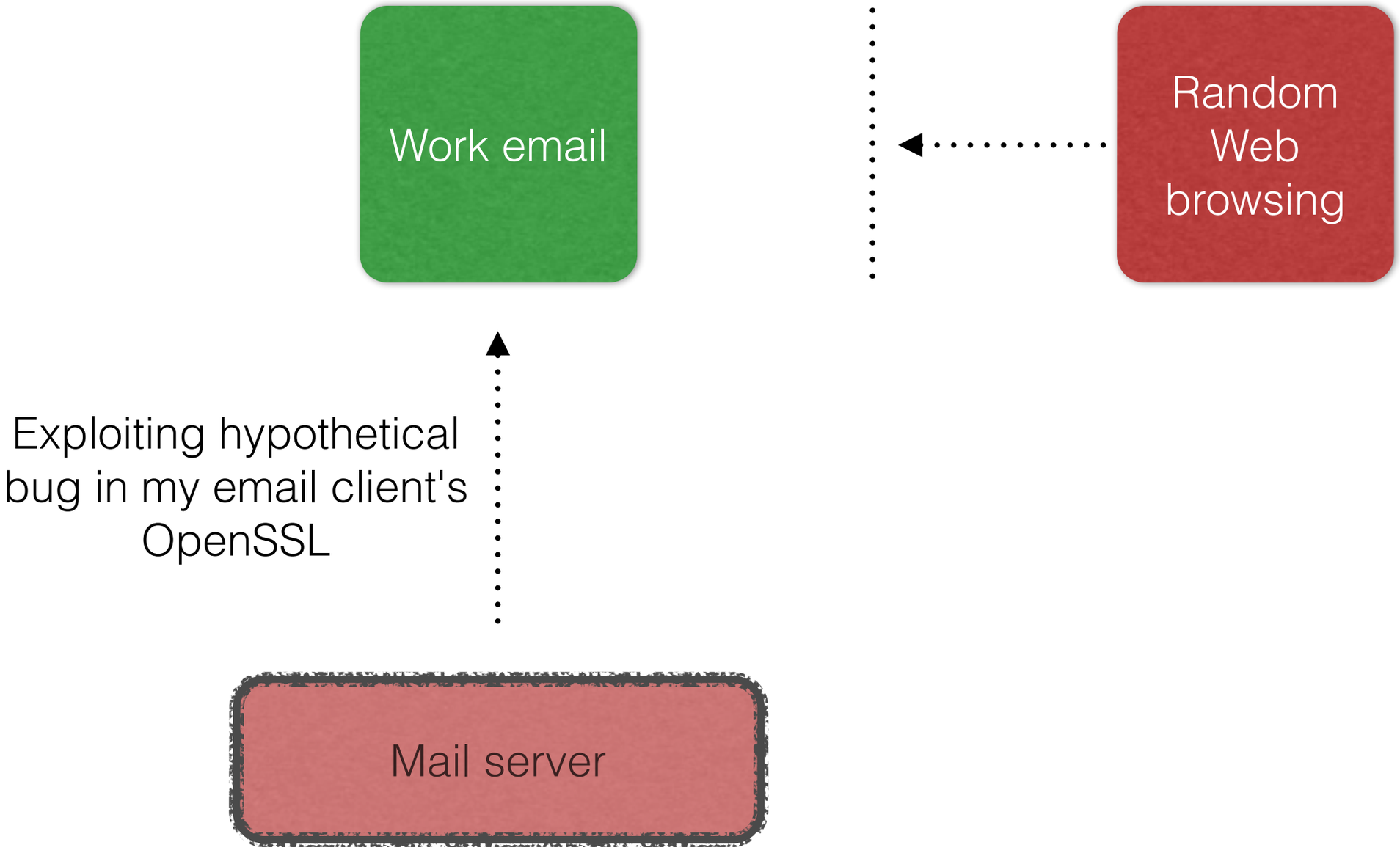


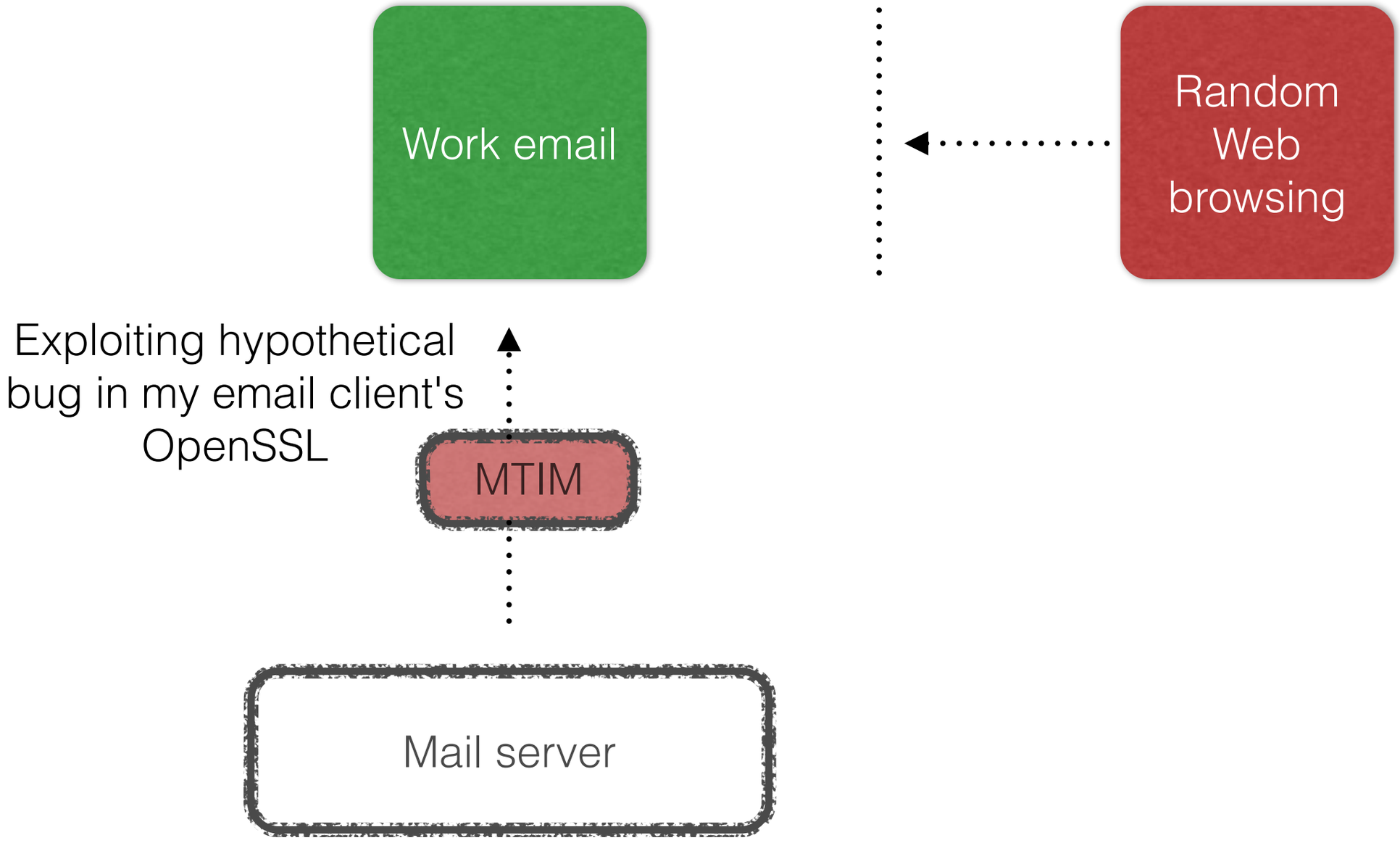
In Qubes we copy files between domains using shared memory and simple cpio-like tool (this cpio-like tool is the security critical code)

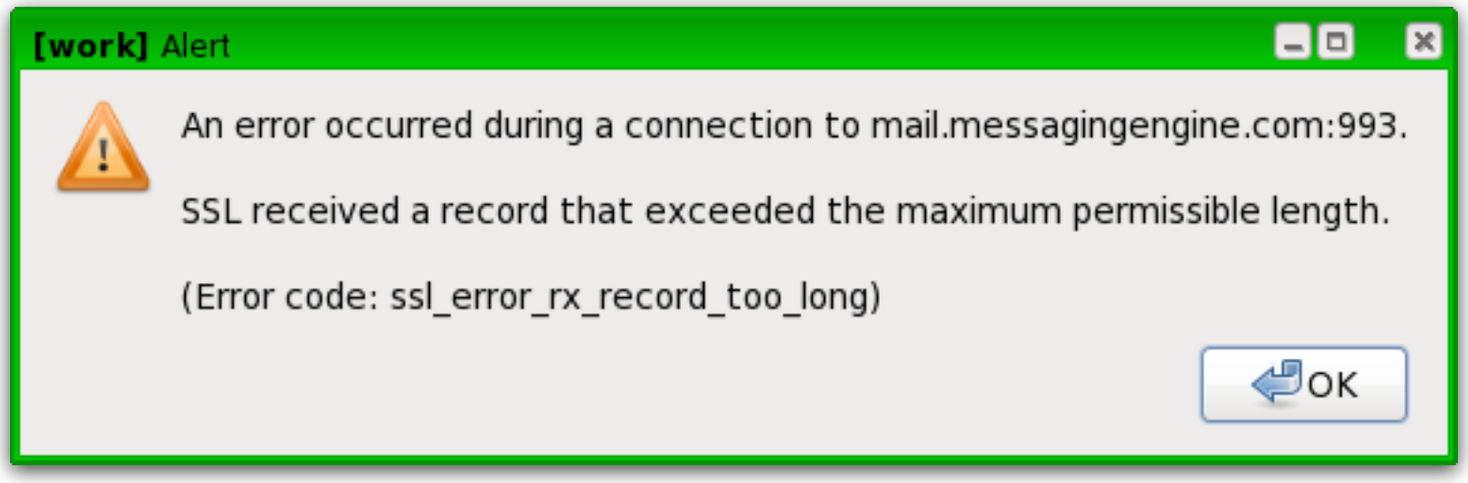
Limitations of Security by Isolation approach

Security by Isolation doesn't protect your apps from being compromised!


Work email
Random
Web
browsing


Work email
Random
Web
browsing
Exploiting hypothetical
bug in my email client's
OpenSSL
Mail server


Work email
Random
Web
browsing
Exploiting hypothetical
bug in my email client's
OpenSSL
MTIM
Mail server



My recent adventure in a hotel in Paris ;)

Solution: decompose the app! (More security by isolation!)

GPG

Email parsing

OpenSSL handling

Capsicum is working on such app-level decompositions
(will definitely use in Qubes when ready)

Another approach: safe languages

(so, where can I get thunderbird-like app written in C#?)

Security by Isolation: Useful technologies

Technologies for **address space isolation**

MMU

Virtualization
(VT-x/AMD-v, EPT/NPT)

Analogies

MMU	VT-x/EPT
User mode (ring 3)	Guest mode (non-root)
Kernel mode (ring 0)	Hypervisor (root mode)
Page Tables	Extended Page Tables (EPT)
Exceptions (#GP, #PF, ...)	VM exits

Differences

MMU	VT-x/EPT
User mode and kernel mode often share the same address space (e.g. 3/1GB split on 32bit Linux)	Guest and the hypervisor never share the same address space
SIPI interrupts kernel execution	SIPI is blocked in VMX

SMEP somehow eliminates this difference

Interrupt Remapping makes this irrelevant anyway

So, why bother using virtualization?

Why not just use the good old MMU for address space isolation?

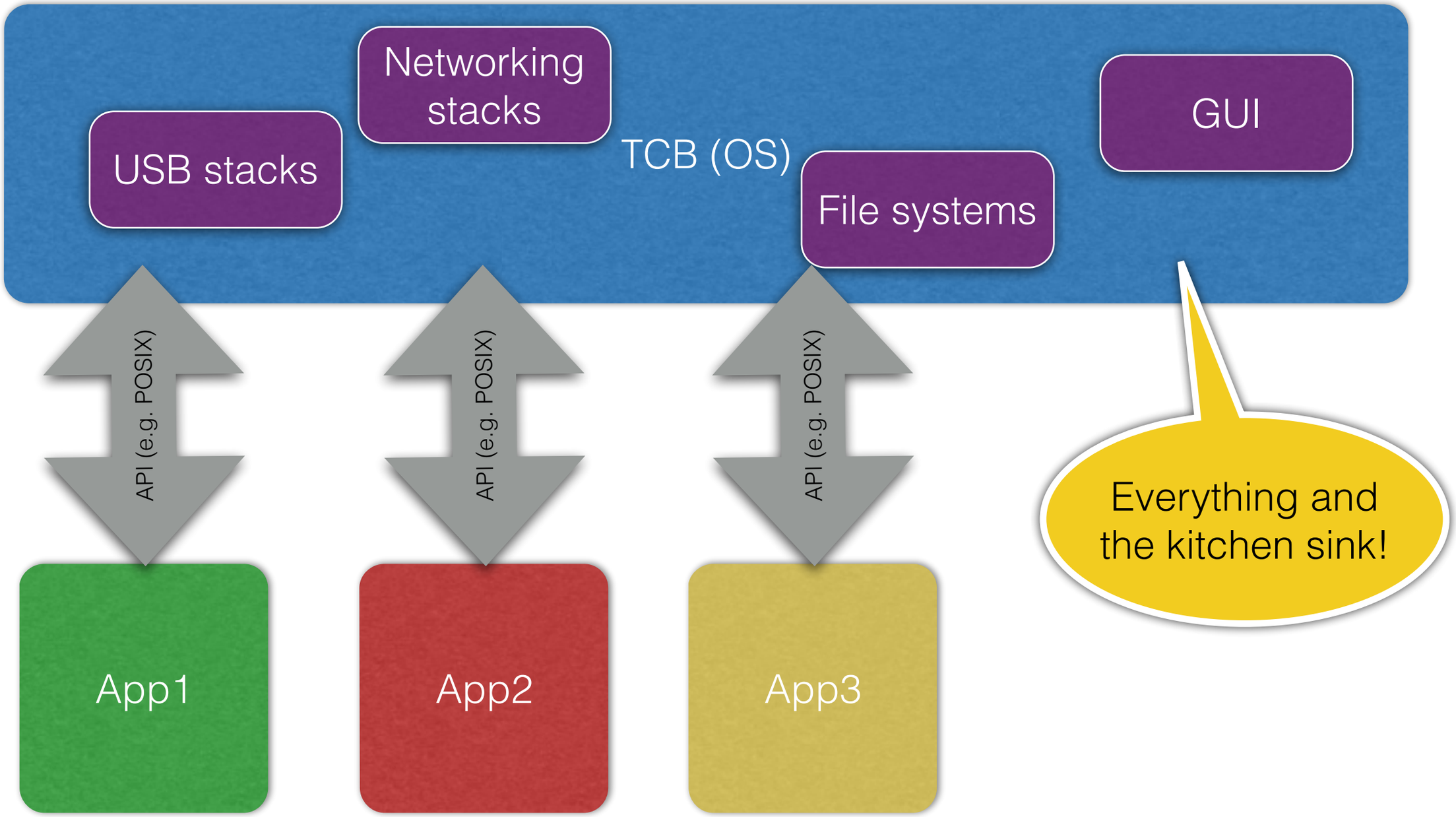
For **compatibility** with OSes that are not para-virtualizable

Linux is PV aware and we can virtualize it using MMU under Xen (Run it as ring3, no need for VT-x)

But why would we want to virtualize the OS in the first place?

A virtualized buggy, messy OS is still... a buggy, messy OS!

Because we want to use the OS as an **API provider**!


Networking stacks
USB stacks
TCB (OS)
File systems
GUI
API (e.g. POSIX)
API (e.g. POSIX)
API (e.g. POSIX)
Everything and the kitchen sink!
App1
App2
App3

CPU scheduling, MMU & IOMMU only

TCB (microkernel/hypervisor)

App1

App2

App3

Backend

Networking Drivers & stacks

Backends

Storage drivers and backends (block, pvusb)

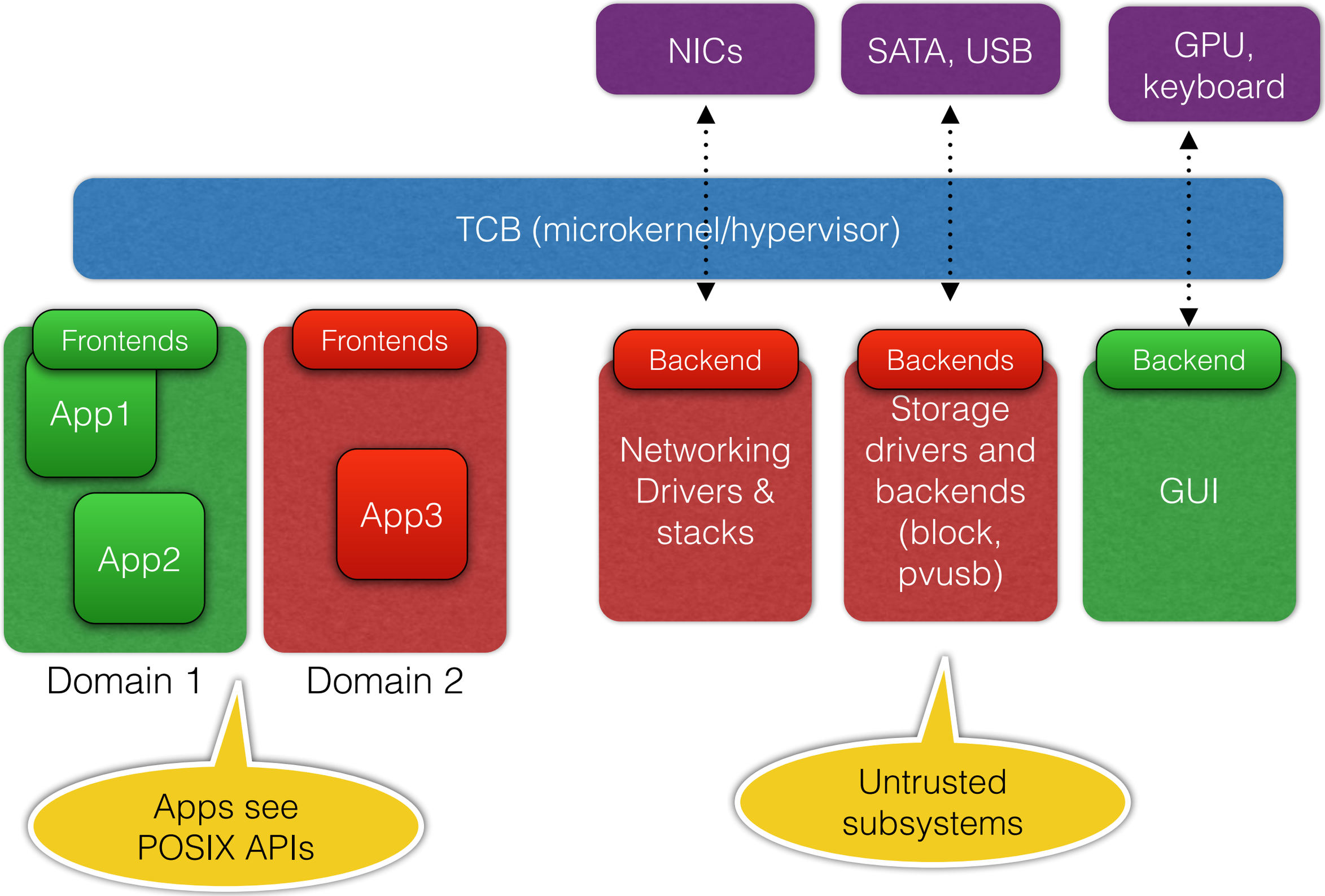
Backend

GUI

But those (legacy) apps expect a POSIX API, they don't know how to talk to the backends

Untrusted subsystems

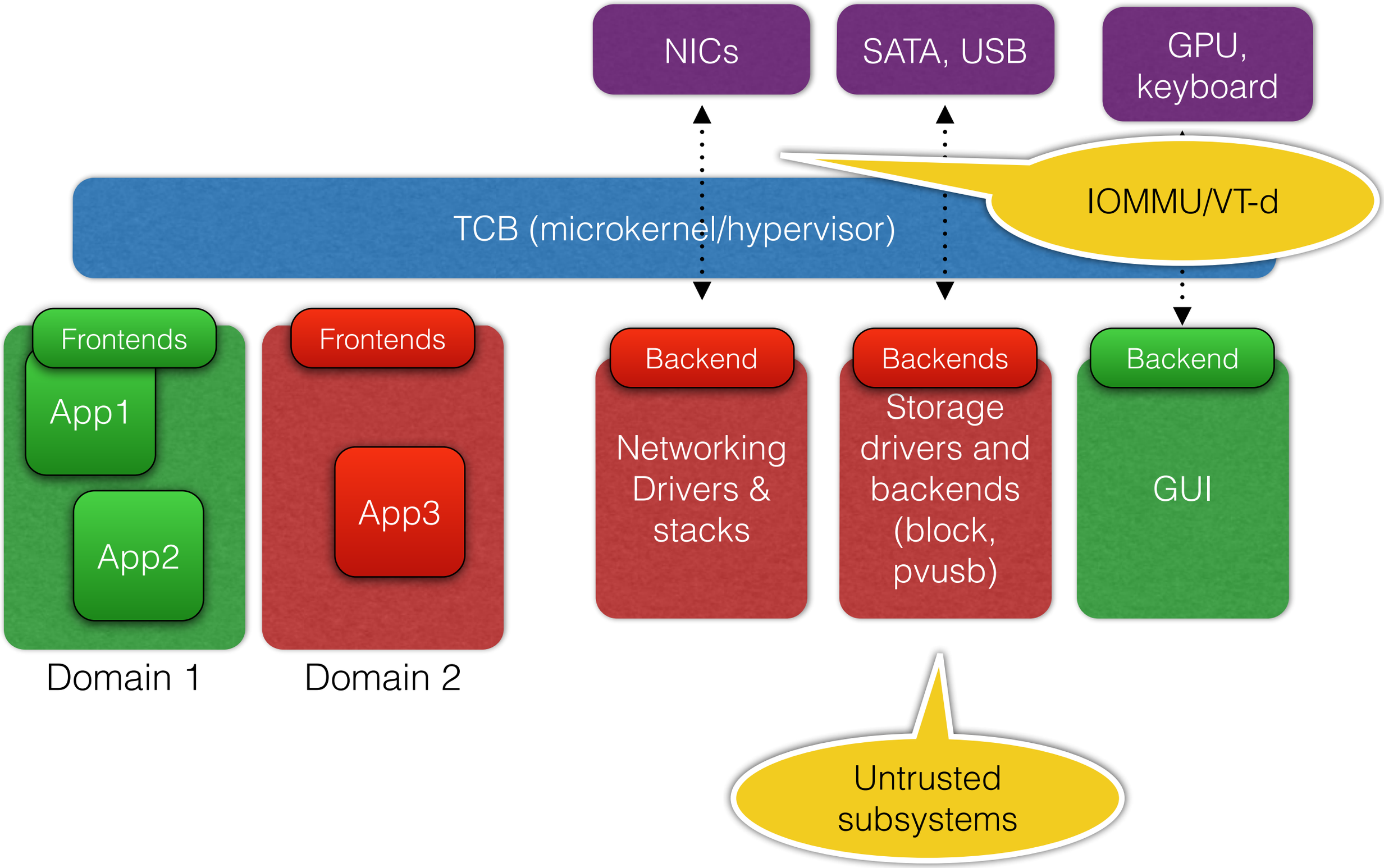
So we must virtualize the whole OS to provide API for legacy apps...


NICs
SATA, USB
GPU, keyboard
TCB (microkernel/hypervisor)
Frontends
App1
App2
Frontends
App3
Backend
Networking Drivers & stacks
Backends
Storage drivers and backends (block, pvusb)
Backend
GUI
Domain 1
Domain 2
Apps see POSIX APIs
Untrusted subsystems

But it is not like virtualization (VT-x) provides stronger security than MMU!

IOMMU (VT-d)

IOMMU allows to sandbox drivers and devices, so plays a key role in TCB disaggregation...


NICs
SATA, USB
GPU, keyboard
TCB (microkernel/hypervisor)
IOMMU/VT-d
Frontends
App1
App2
Frontends
App3
Backend
Networking Drivers & stacks
Backends
Storage drivers and backends (block, pvusb)
Backend
GUI
Domain 1
Domain 2
Untrusted subsystems

IOMMU: catches

For safe language-based OSes (e.g. Singularity and derivatives) IOMMU is needed to restrict devices to accesses to their DMA buffers only to preserve memory safety

Catches:

- MSI attacks
- BDF Spoofing
- Reflashing device firmware?

Interrupt Remapping
(see our latest paper on VT-d escapes)

PCIe ACS

DMA-resistant
trusted boot

We really need more trusted trusted boots!
(subject for another presentation)

No secure client systems without IOMMU and trusted boot!

Security by Isolation: Challenges

How to partition my digital life into security **domains**?

Do we actually need domains? Perhaps we can just isolate each **app** from each other app?

We need OSes to provide legacy APIs to apps

Would be a waste of memory to have one instance of an OS per each app...

But even if we did isolate (virtualize?) on a per app granularity, still the problem of partitioning doesn't go away...

Mail

Mail Personal

Mail Work

Unless we get 100% safe languages we would not avoid security by isolation...

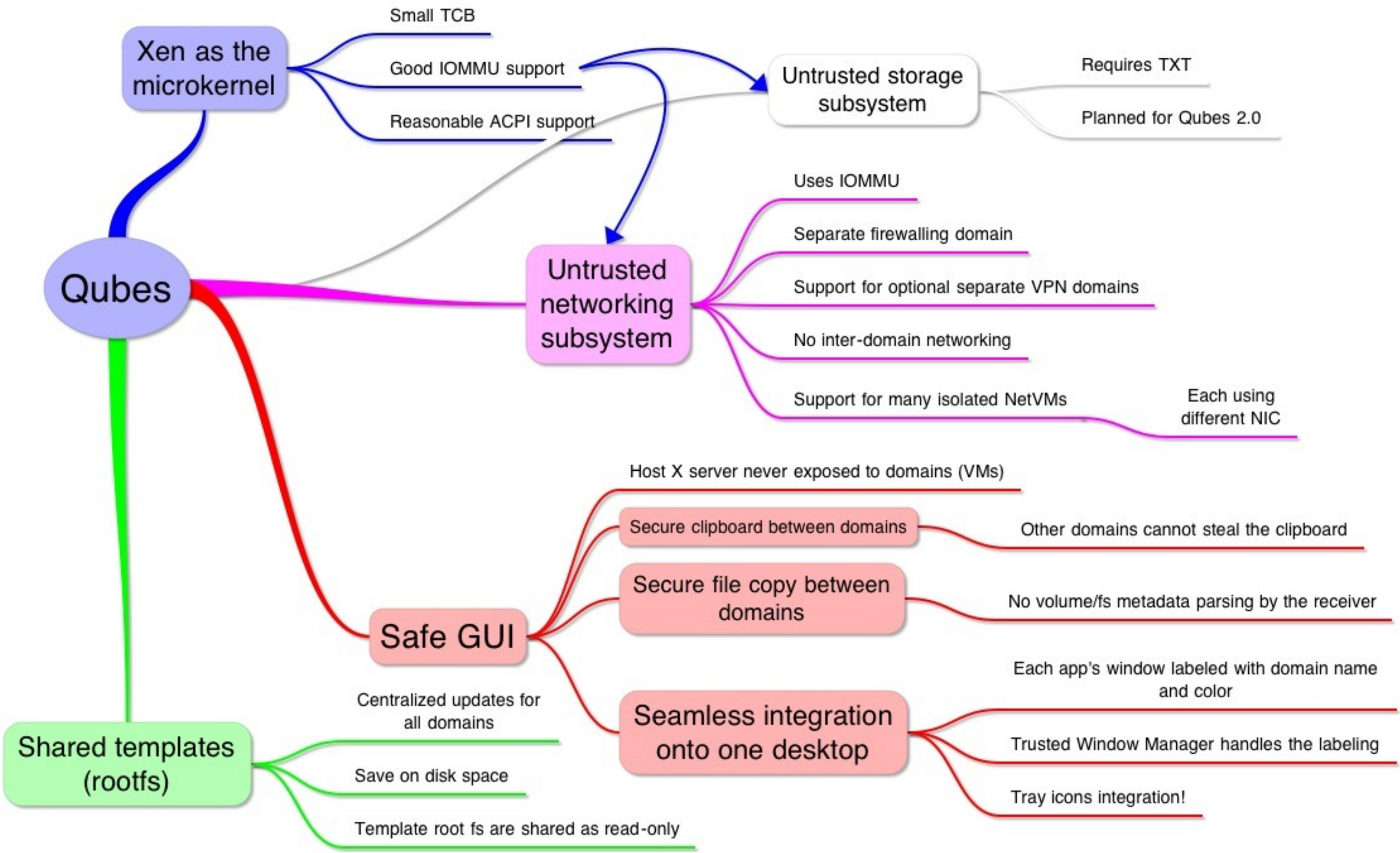
Other challenges

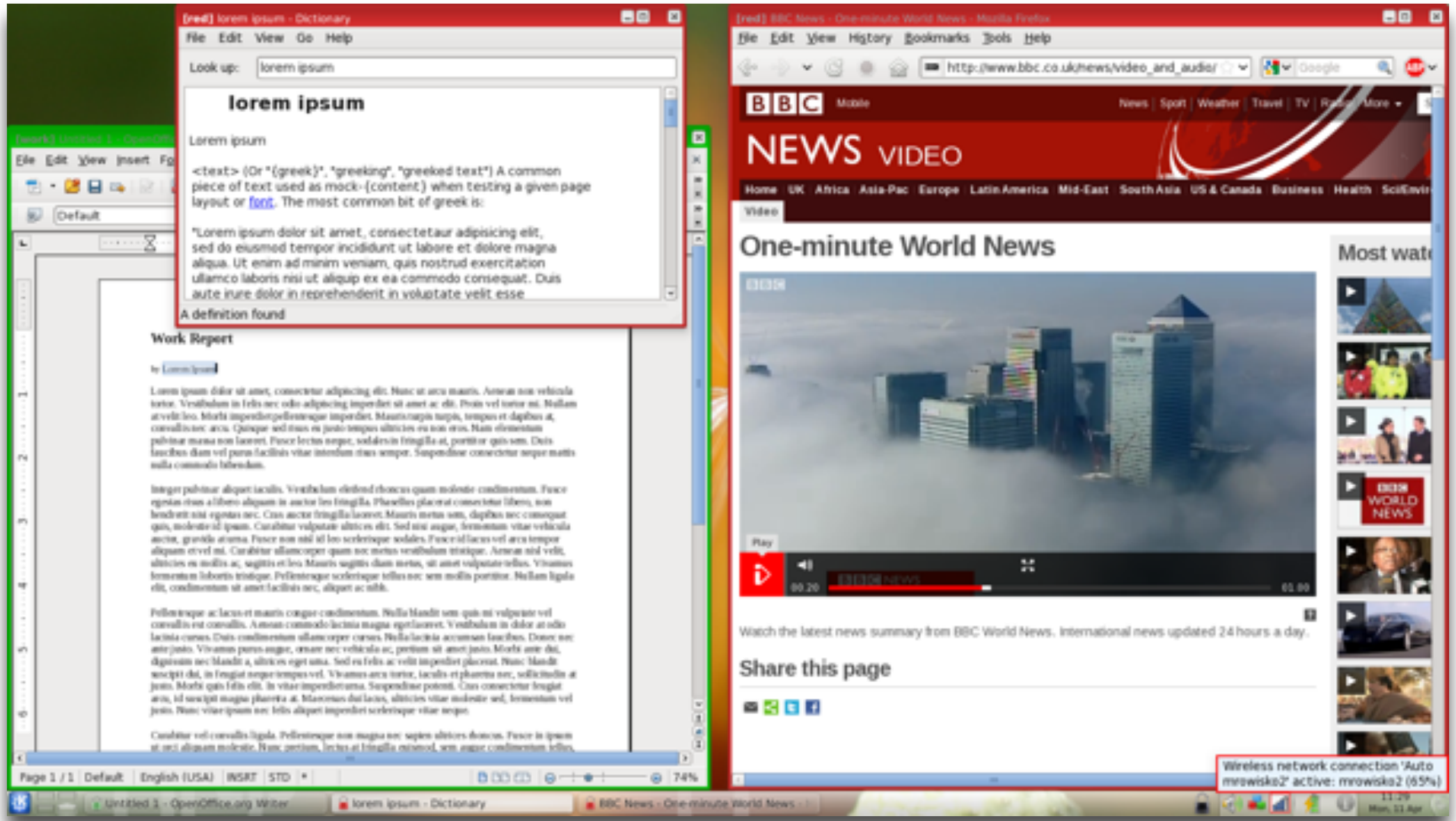
GPU multiplexing

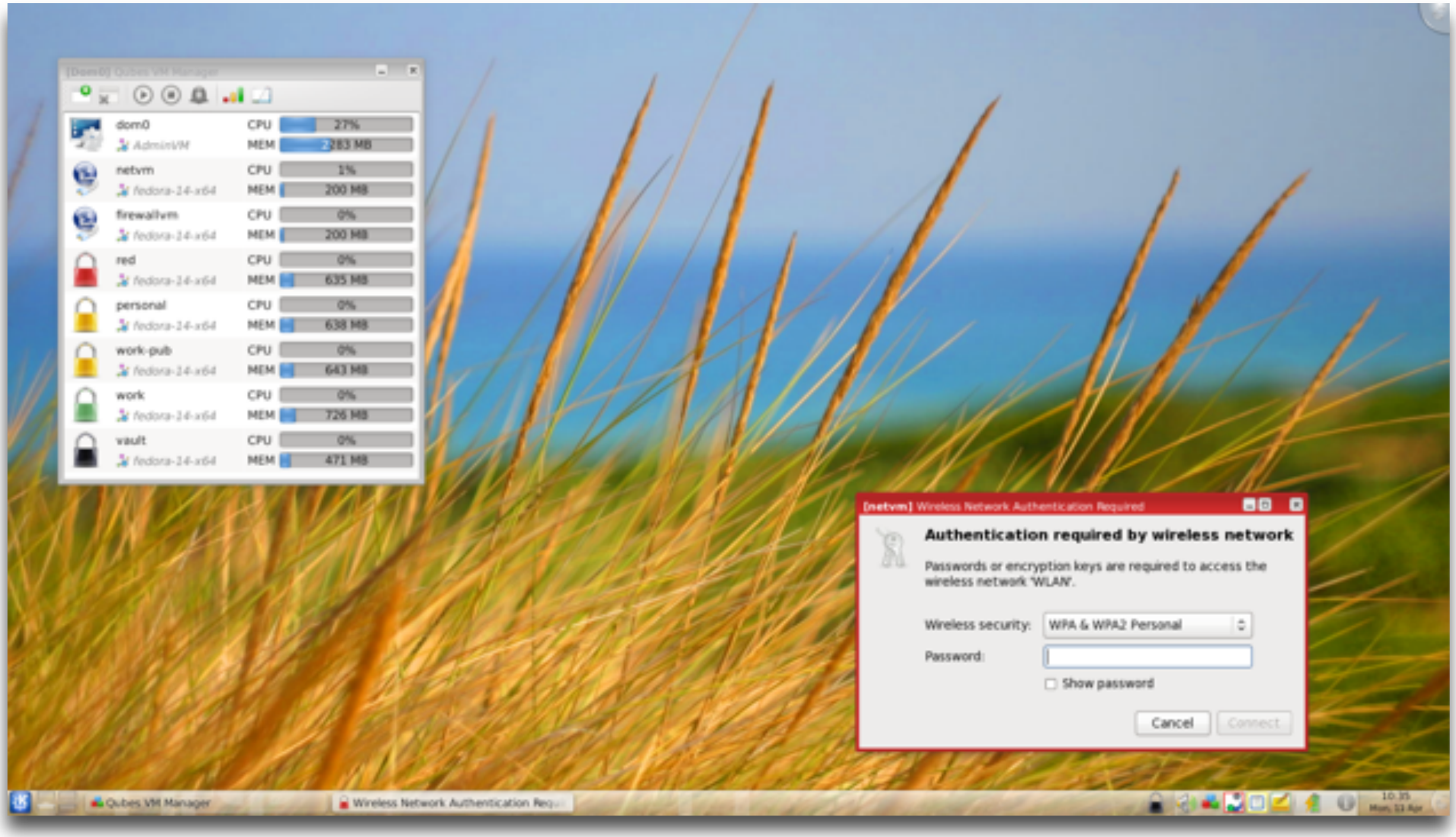
USB multiplexing

I'd love to discuss that last two problems!

Qubes OS implements lots of ideas mentioned here


Xen as the microkernel
Small TCB
Good IOMMU support
Reasonable ACPI support
Untrusted storage subsystem
Requires TXT
Planned for Qubes 2.0
Qubes
Untrusted networking subsystem
Uses IOMMU
Separate firewalling domain
Support for optional separate VPN domains
No inter-domain networking
Support for many isolated NetVMs
Each using different NIC
Host X server never exposed to domains (VMs)
Safe GUI
Secure clipboard between domains
Other domains cannot steal the clipboard
Secure file copy between domains
No volume/fs metadata parsing by the receiver
Seamless integration onto one desktop
Each app's window labeled with domain name and color
Trusted Window Manager handles the labeling
Tray icons integration!
Shared templates (rootfs)
Centralized updates for all domains
Save on disk space
Template root fs are shared as read-only


[red] lorem ipsum - Dictionary
File Edit View Go Help
Look up:
lorem ipsum
lorem ipsum
Lorem ipsum
A definition found
Work Report
[red] BBC News - One-minute World News - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.bbc.co.uk/news/video_and_audio/
BBC Mobile
NEWS VIDEO
Home UK Africa Asia-Pac Europe Latin America Mid-East South Asia US & Canada Business Health
Video
One-minute World News
Most wate
Play
Watch the latest news summary from BBC World News. International news updated 24 hours a day.
Share this page
Page 1 / 1 Default English (USA) INSRT STD
74%
Wireless network connection 'Auto mrowisko2' active: mrowisko2 (65%)
Untitled 1 - OpenOffice.org Writer
lorem ipsum - Dictionary
BBC News - One-minute World News -


[Dom0] Qubes VM Manager
dom0
AdminVM
CPU 27%
MEM 2283 MB
netvm
fedora-14-x64
CPU 1%
MEM 200 MB
firewallvm
fedora-14-x64
CPU 0%
MEM 200 MB
red
fedora-14-x64
CPU 0%
MEM 635 MB
personal
fedora-14-x64
CPU 0%
MEM 638 MB
work-pub
fedora-14-x64
CPU 0%
MEM 643 MB
work
fedora-14-x64
CPU 0%
MEM 726 MB
vault
fedora-14-x64
CPU 0%
MEM 471 MB
[netvm] Wireless Network Authentication Required
Authentication required by wireless network
Passwords or encryption keys are required to access the wireless network 'WLAN'.
Wireless security:
WPA & WPA2 Personal
Password:
Show password
Cancel
Connect
Qubes VM Manager
Wireless Network Authentication Requ...

Qubes is not a microkernel....

... It's everything else!

Qubes-OS.org

Thanks!