Isolation

Mustakimur R. Khandaker

Running Untrusted Code

We often need to run **buggy/untrusted code**:

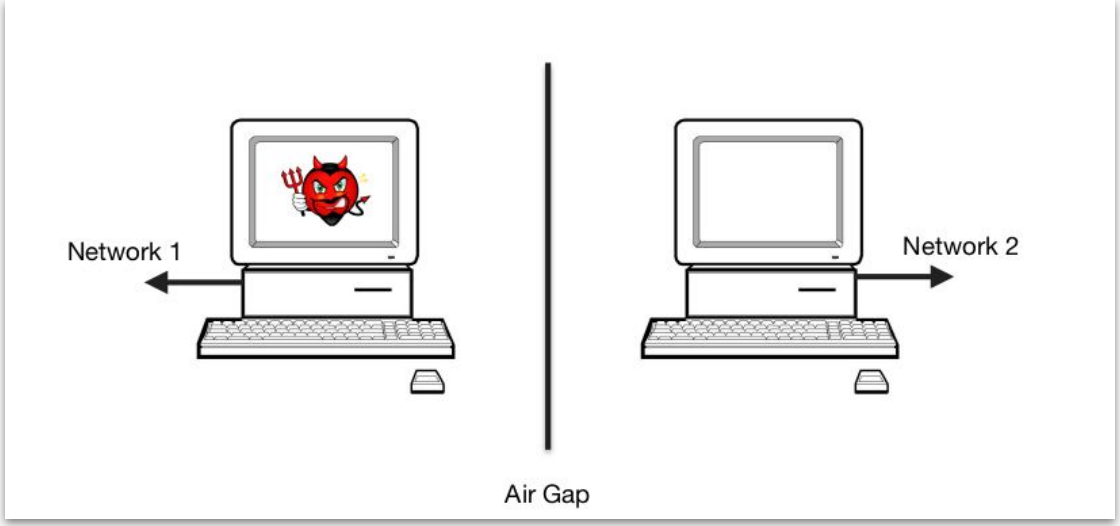
- programs from untrusted Internet sites:
  apps, extensions, plug-ins, codecs for media player.
- exposed applications:
  pdf viewers, outlook.
- legacy daemons: sendmail, bind.
- Honeypots.

Goal: if untrusted code "misbehaves" ➡ kill it.

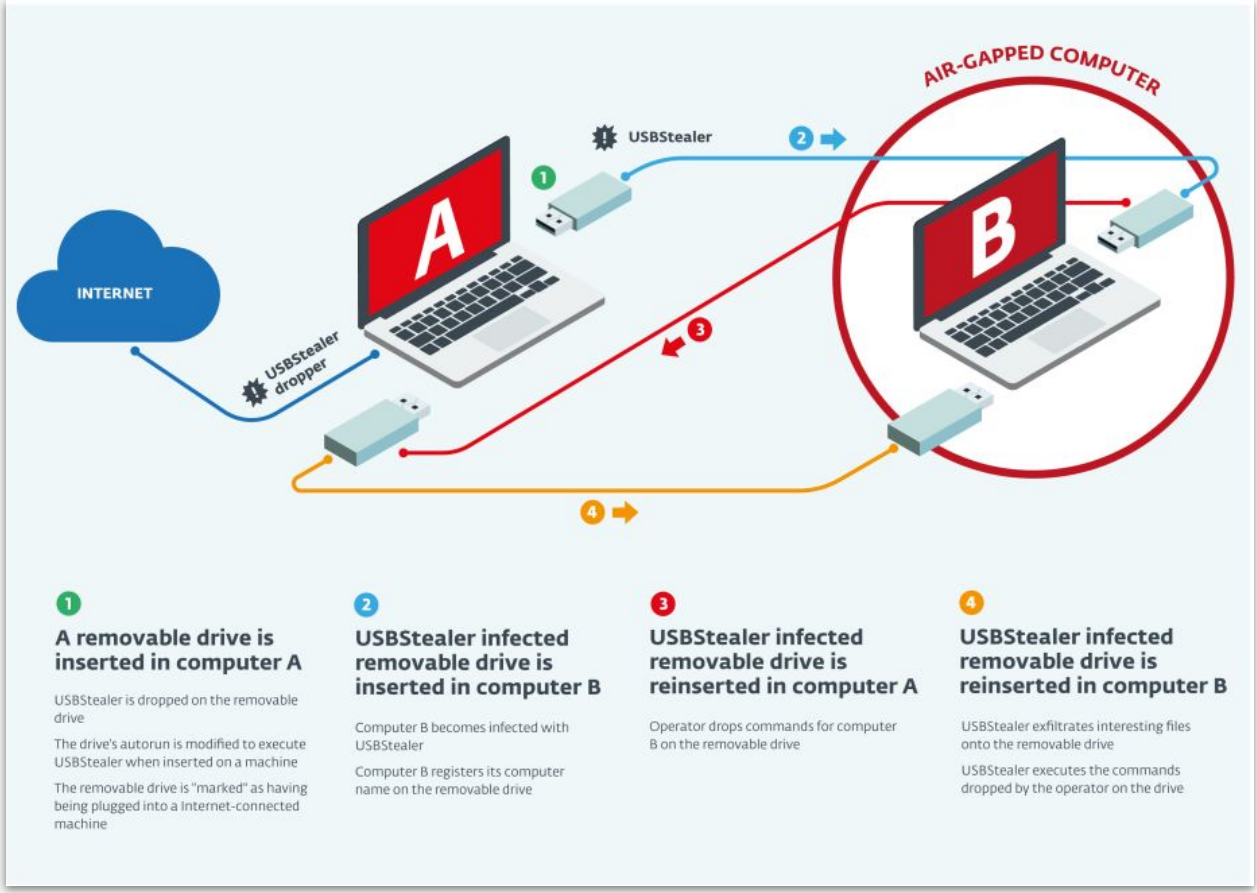
Approach: Confinement

Confinement: ensure misbehaving code cannot harm rest of system.

- Confinement can be implemented at many levels.
  - **Hardware:** run untrusted code on isolated hardware (air gap).
    - e.g., root certificate management.
    - Difficult to use and manage.



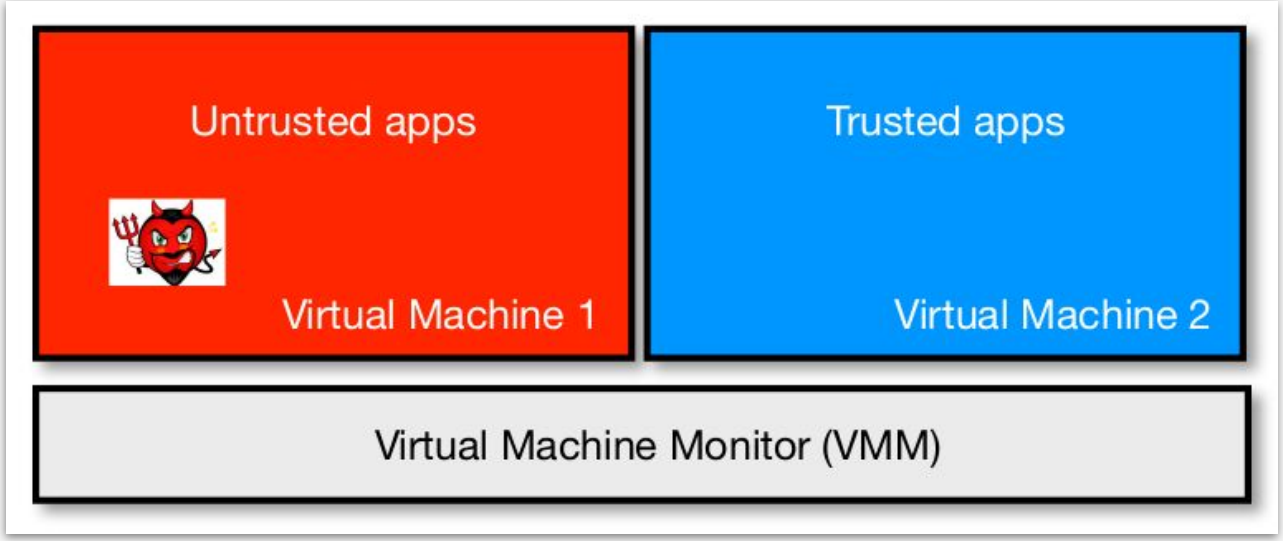
Sidenote: Attacking Air-gapped Computers



Approach: Confinement

Confinement: ensure misbehaving code cannot harm rest of system.

- Confinement can be implemented at many levels.
  - Virtual machines: isolate OS's on a single machine.

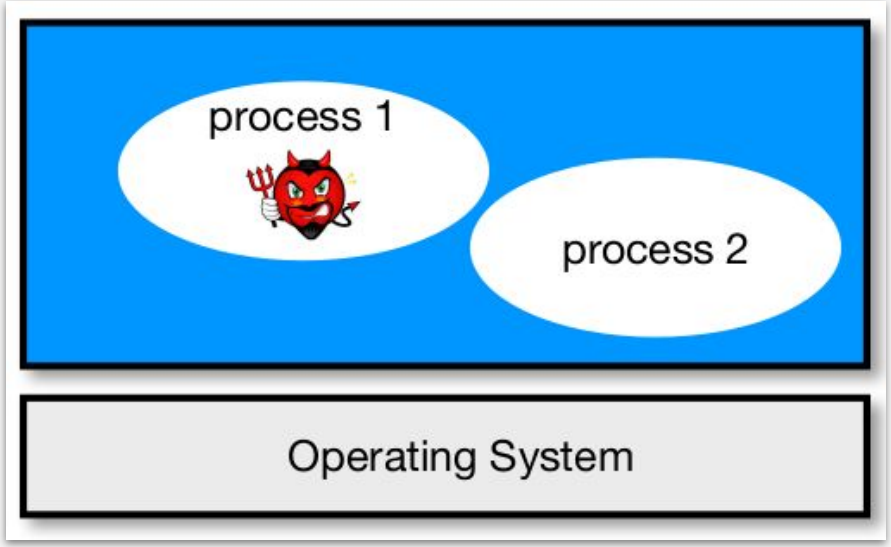


Approach: Confinement

Confinement: ensure misbehaving code cannot harm rest of system

- Confinement can be implemented at many levels.
  - **Process:** isolate a process in a single operating system.

  Example: android user-based app isolation.



Approach: Confinement

Confinement: ensure misbehaving code cannot harm rest of system

- Confinement can be implemented at many levels
  - **Threads:** isolate threads sharing the same process address space.
    Software fault isolation (SFI)
  - Example: Native Client.

Implementing Confinement

Key component: **reference monitor.**

- **Mediates requests** from applications.
  Implements protection policy.
  Enforces isolation and confinement.
- Must **always** be invoked:
  Every application request must be mediated.
- **Tamperproof:**
  Reference monitor cannot be killed.
  ... or if killed, then monitored process is killed too.
- **Small** enough to be analyzed and validated.

Principle of Least Privilege

What's a privilege?
- Ability to access or modify a resource.

Assume compartmentalization and isolation.
- Separate the system into isolated compartments.
- Limit interaction between compartments.

Principle of Least Privilege
- A system module should only have the minimal privileges needed for its intended purposes.

An Old Example: Chroot

- **Chroot** is often used for *"guest"* accounts on ftp sites.
- To use: (must be root).

```
chroot /tmp/guest
su guest
```

- Root dir "/" is now "/tmp/guest".
- EUID set to *"guest"*.
- "/tmp/guest" is prefixed to any **file system accesses** by applications in jail; applications cannot access files out of jail.

  open ("/etc/passwd", "r") ➡ **open** ("/tmp/guest/etc/passwd", "r")
- **Other system resources are shared /w non-chroot processes.**

Jailkit

Jailkit

Problem: all utility programs (ls, ps, vi) must live inside jail.

- jailkit project: auto builds files, libs, and dirs needed in jail env.
  - **jk_init:** creates jail environment.
  - **jk_check:** checks jail env for security problems.
    checks for any modified programs,
    checks for world writable directories, etc.
  - **jk_lsh:** restricted shell to be used inside jail

Note: simple chroot jail does **not** limit network access.

Escaping from Jails

Early escapes: relative paths.

open(“../../etc/passwd”, “r”) ➡

open(“/tmp/guest/../../etc/passwd”, “r”)

Chroot should only be executable by **root**, otherwise jailed app could become root by (real bug in Ultrix 4.0).

- create dummy file “/aaa/etc/passwd”.
- run chroot “/aaa”.
- run su root to become root.

Many Ways to Escape Jail as Root

- Create device that lets you access raw disk.
- Send signals to non chrooted process.
- Reboot system.
- Bind to privileged ports.
- ...

FreeBSD Jail

Stronger mechanism than simple chroot.

- Can be considered as operating system virtualization.

To run: ***jail jail-path hostname IP-addr cmd***

- Calls hardened chroot (no "../../" escape).
- Can only bind to sockets with specified IP address and authorized ports.
- Can only communicate with processes inside jail.
- Root is limited, e.g. cannot load kernel modules.

Problems with Chroot and Jail

Coarse policies:

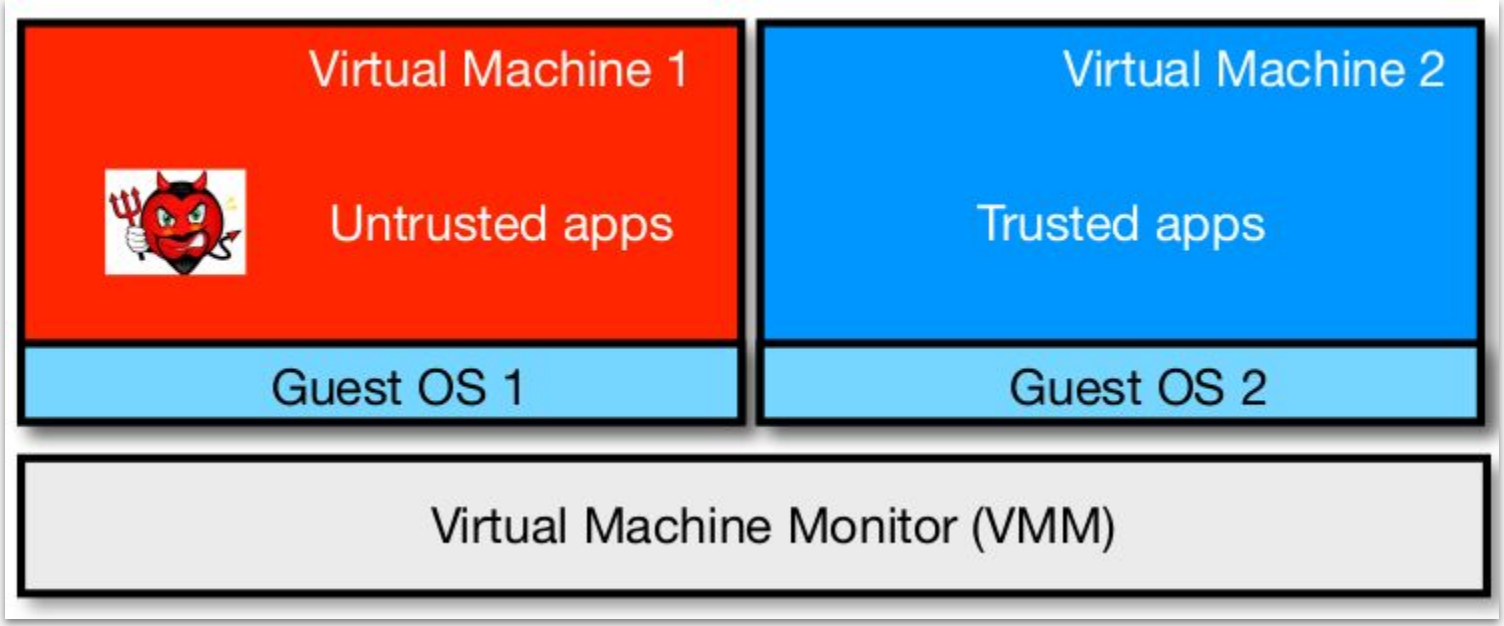
- All or nothing access to parts of file system.
- Inappropriate for apps like a web browser.
  Needs read access to files outside jail.
  (e.g. for sending attachments in Gmail)

Poor isolation, does not prevent malicious apps from:

- Accessing network and messing with other machines.
- Trying to crash host OS.

Virtual Machines

Virtual Machines



Single HW platform used for both trusted and untrusted apps.

History of Virtualization

VMs in the **1960**'s:
- Few computers, lots of users.
- VMs allow many users to shares a single computer.

VMs **1970**'s – **2000**: non-existent.

VMs since **2000**:
- Too many computers, too few users.
  Print server, Mail server, Web server, File server, Database , ...
- Wasteful to run each service on different hardware.
- More generally: VMs heavily used in cloud computing.

VMM Security Assumption

VMM security assumption:

- Malware can infect guest OS and guest apps
- But malware cannot escape from the infected VM
  - Cannot infect host OS
  - Cannot infect other VMs on the same hardware

Requires that VMM protect itself and is not buggy.

- VMM is much simpler than full OS ... but device drivers run in Host OS.

VM escape attacks exist.

- Jail break from virtual machine to VMM/host OS.

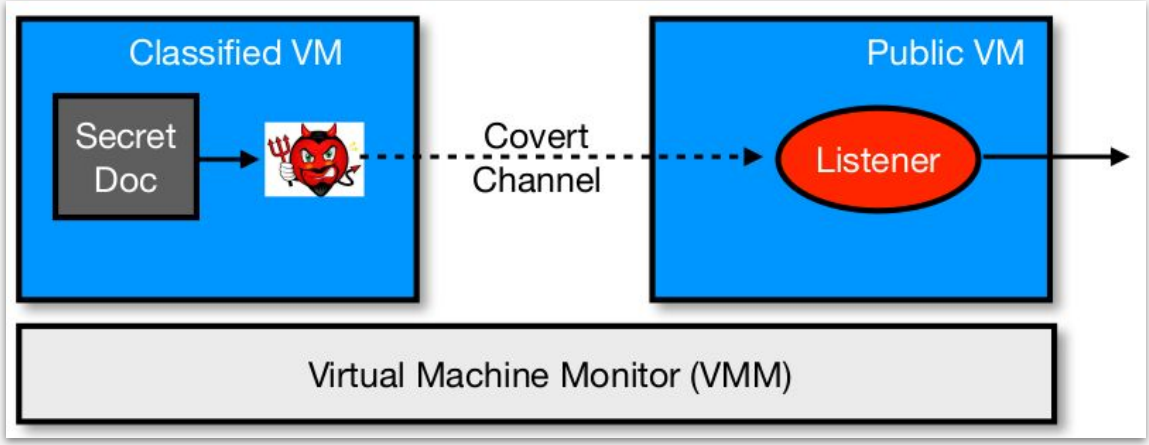
Covert and Side Channels

Covert channel: unintended communication channel between isolated components.

- Colluding malware in the trusted VM.

Side channel: obtain information (e.g., crypto keys) from physical implementation (e.g., shared cache).

- No colluding malware required.



An Example Covert Channel

Both VMs use the same underlying hardware.

To send a bit **b ∈ {0,1}**

- Malware does:
  b = 1: at 1:00am do CPU intensive calculation.
  b = 0: at 1:00am do nothing.
- At 1:00am **listener** does CPU intensive calc. and measures completion time.
  b = 1 ⇔ completion-time > threshold

Many covert channels exist in running system:

- File lock status, cache contents, interrupts, …
- Difficult to eliminate all.

Example: Qubes OS

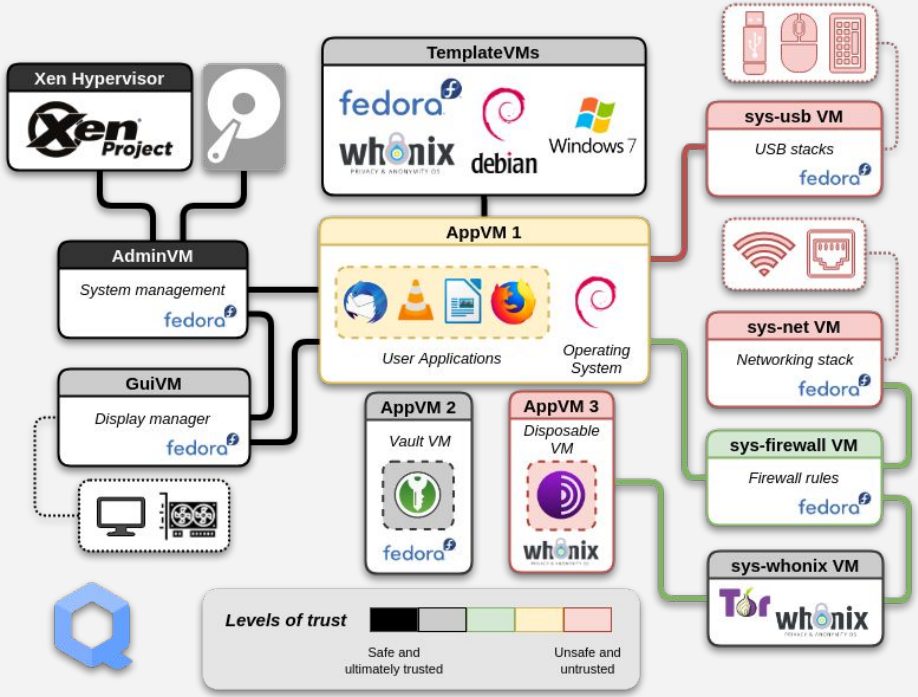
Source: https://www.qubes-os.org

Qubes OS is a free and open-source, security-oriented operating system for single-user desktop computing.

- Qubes OS leverages Xen-based virtualization to allow for the creation and management of isolated compartments called qubes.

Purposes: with a predefined set of one or many isolated applications, for personal or professional projects.

Natures: full-fledged or stripped-down virtual machines based on popular operating systems.

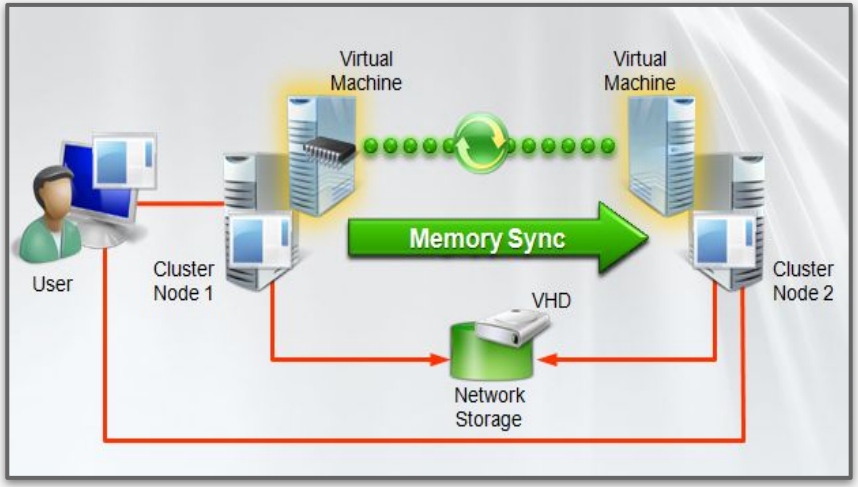


Levels of trust: All windows are displayed with unforgeable colored window borders so that different security levels are easily identifiable.

Live Migration

Live Migration: a process of moving a running virtual machine between different physical machines without disconnecting the client.

Necessity of powerful spare servers to temporarily hold the migrating VMs.

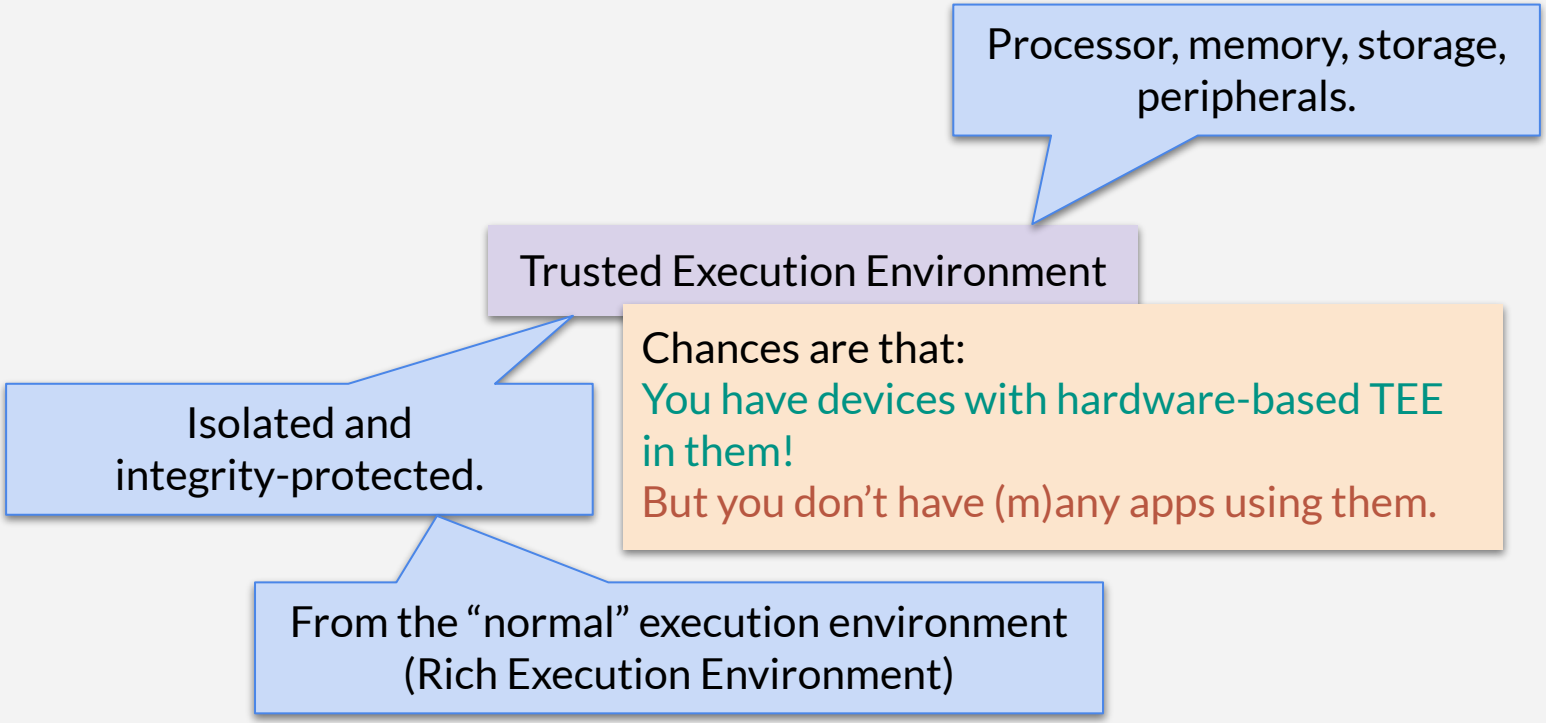


Copy large amounts of memory between VMs without performance degrade.

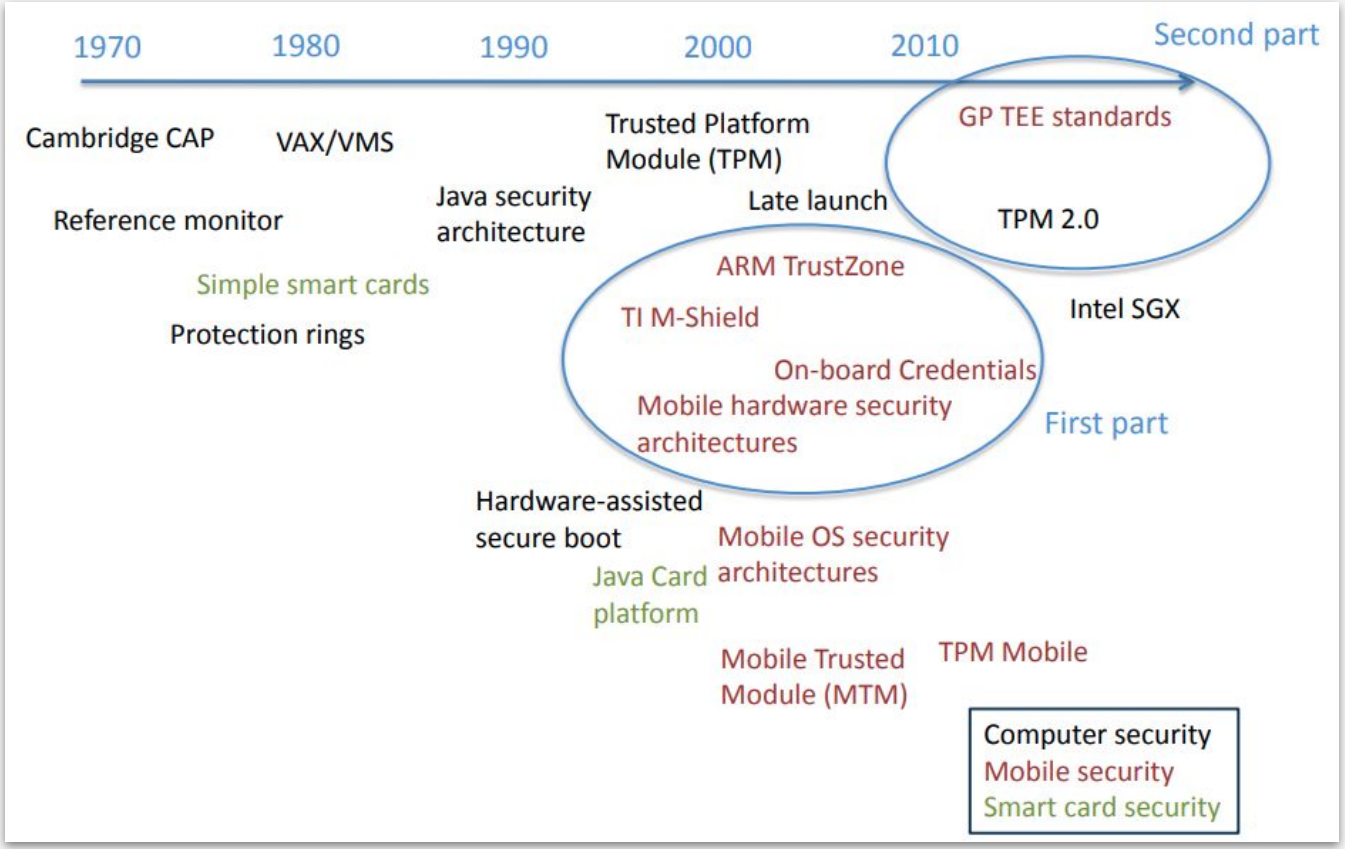
<u>Goal:</u>
design high-end solution of live migration that could be widely adopted by cloud providers e.g. Amazon and Alibaba?

Trusted Execution Environment

What is TEE?

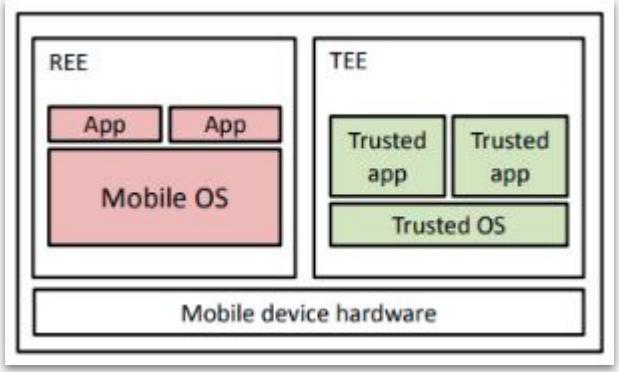

Processor, memory, storage, peripherals.
Trusted Execution Environment
Isolated and integrity-protected.
Chances are that:
You have devices with hardware-based TEE in them!
But you don't have (m)any apps using them.
From the "normal" execution environment (Rich Execution Environment)

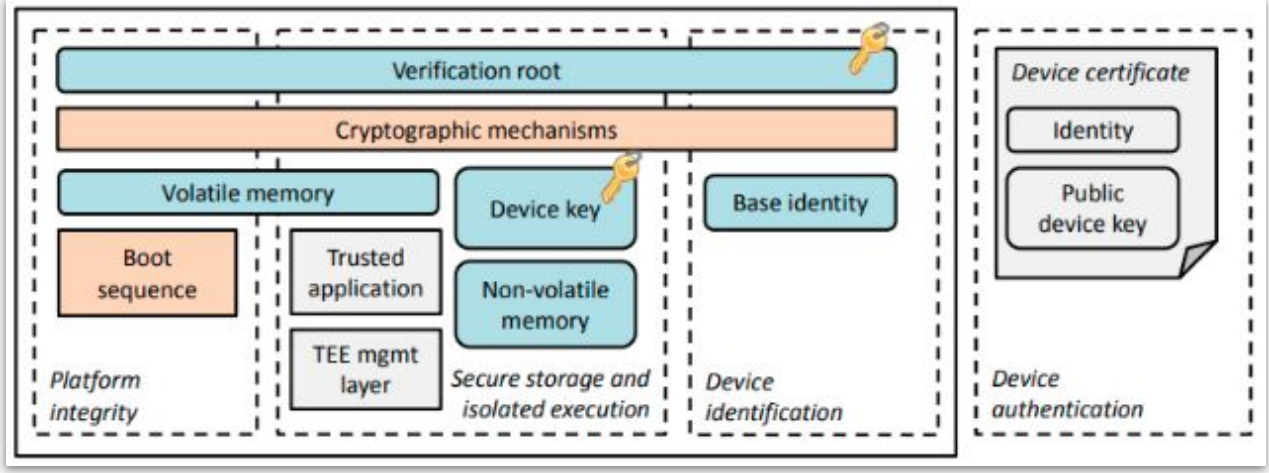
Historical Perspective



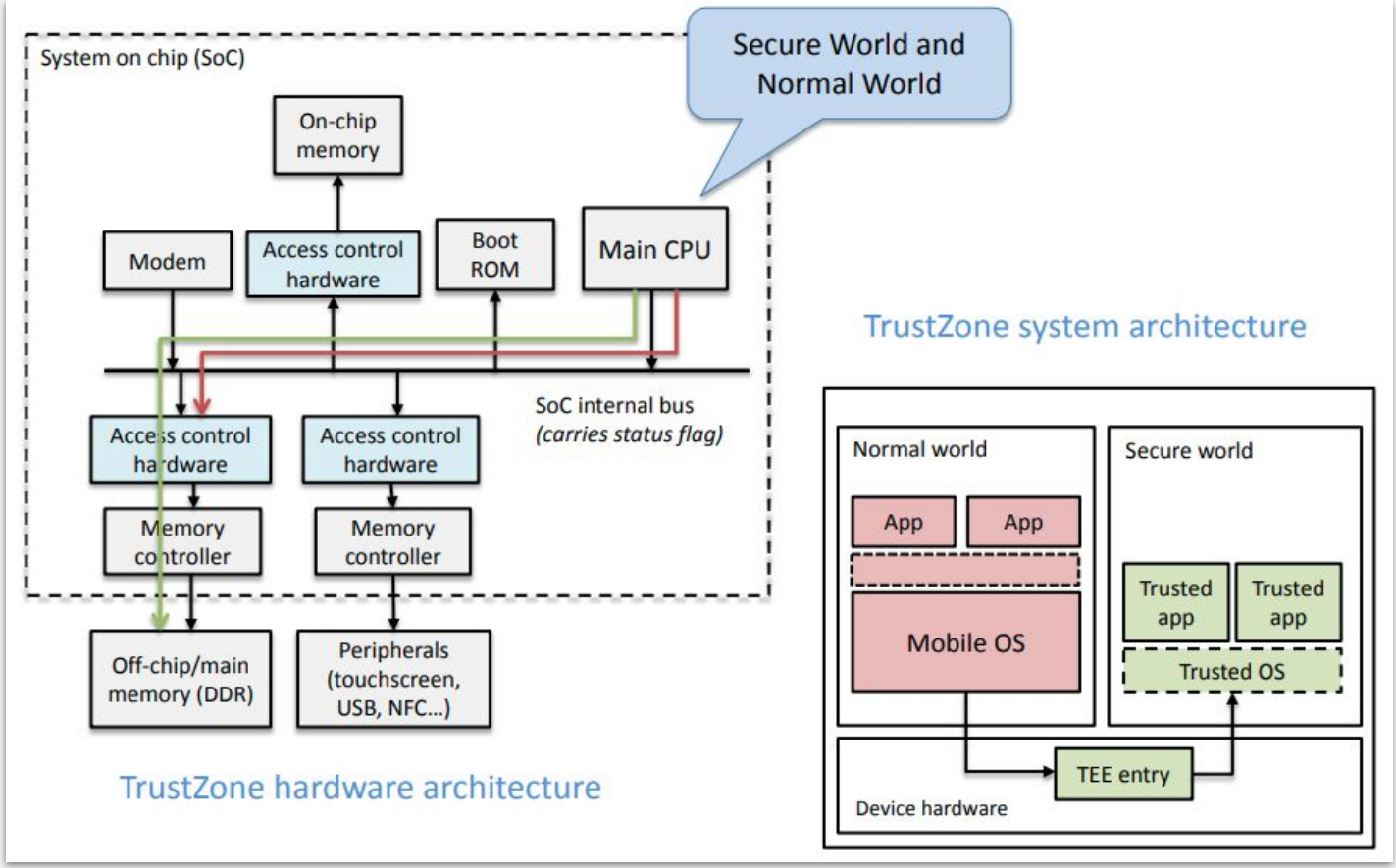
Overview

1. Platform integrity.
2. Secure storage.
3. Isolated execution.
4. Device identification.
5. Device authentication.





ARM TrustZone



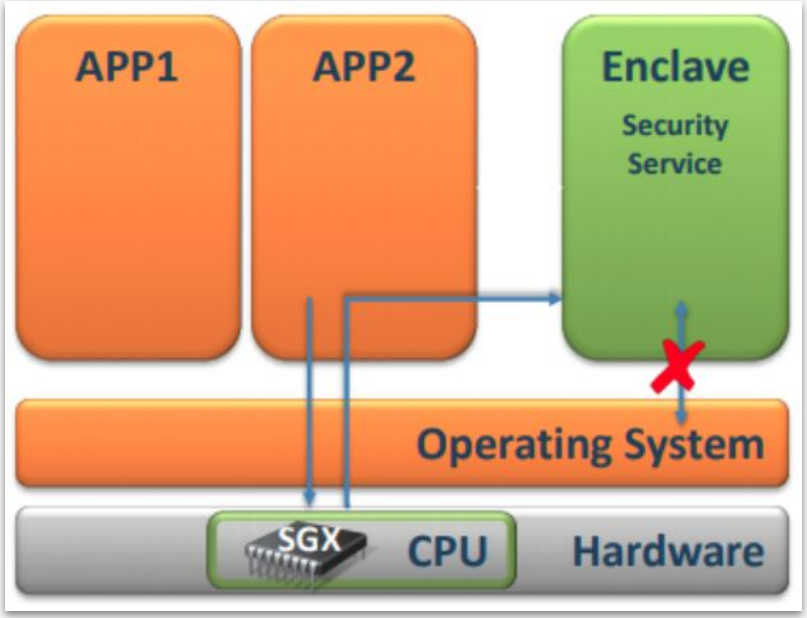
Intel SGX

Security critical code isolated in enclave.

Only CPU is trusted.
- Transparent memory encryption.
- 18 new instructions.

Enclaves cannot harm the system.
- Only unprivileged code (CPU ring3).
- Memory protection.

Designed for multi-core systems.
- Multi-threaded execution of enclaves.
- Parallel execution of enclaves and untrusted code.
- Enclaves are interruptible.



< Isolation />