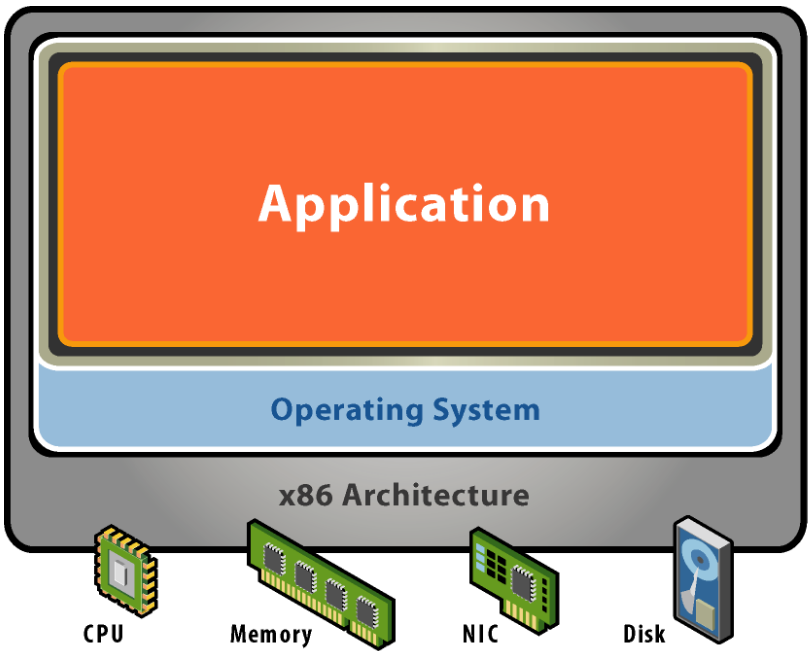
Virtualization Security

Vitaly Shmatikov

Physical Machine



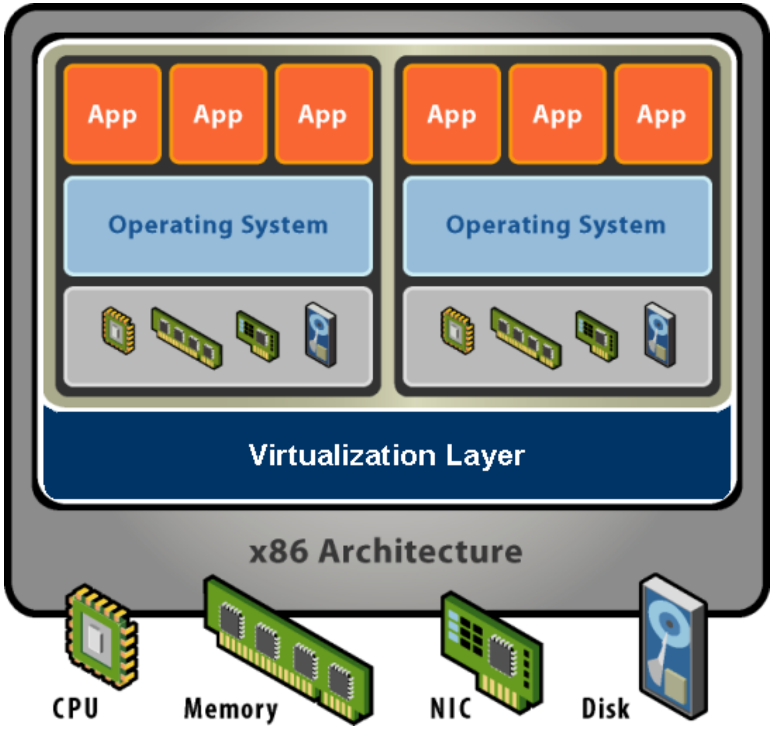
Physical hardware

- Processors, memory, chipset, I/O devices, etc.
- Resources often grossly underutilized

Software

- Tightly coupled to physical hardware
- Single active OS instance
- OS controls hardware

Virtual Machine



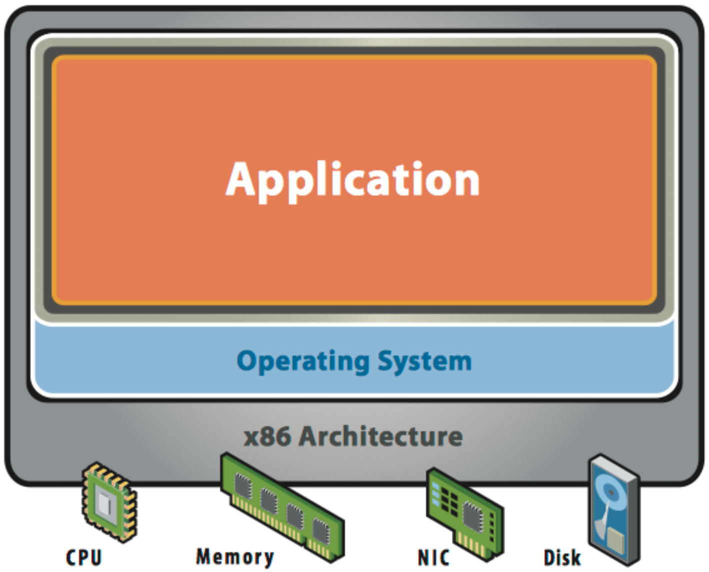
Software abstraction

- Behaves like hardware
- Encapsulates all OS and application state

Virtualization layer

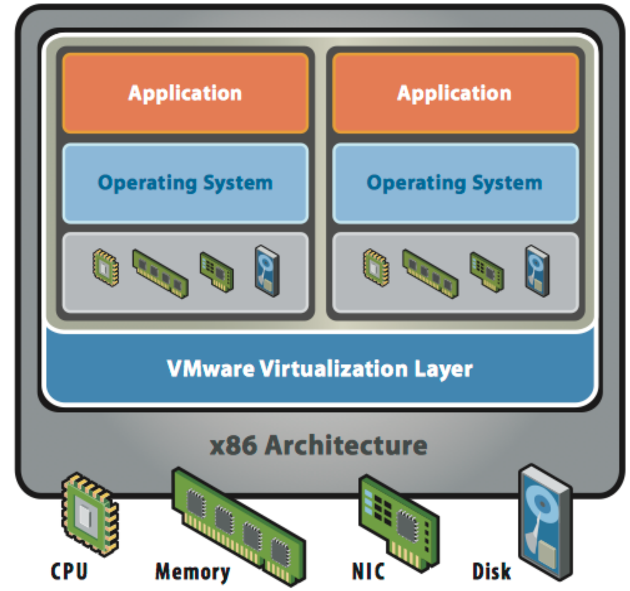
- Extra level of indirection
- Decouples hardware, OS
- Enforces isolation
- Multiplexes physical hardware across VMs

Virtualization



Before Virtualization:

- Single OS image per machine
- Software and hardware tightly coupled
- Running multiple applications on same machine often creates conflict
- Underutilized resources
- Inflexible and costly infrastructure



After Virtualization:

- Hardware-independence of operating system and applications
- Virtual machines can be provisioned to any system
- Can manage OS and application as a single unit by encapsulating them into virtual machines

Types of Virtualization

Process virtualization
- Language-level   Java, .NET, Smalltalk
- OS-level processes, Solaris Zones, BSD Jails, Virtuozzo
- Cross-ISA emulation Apple 68K-PPC-x86, Digital FX!32

Device virtualization
- Logical vs. physical VLAN, VPN, NPIV, LUN, RAID

System virtualization
- “Hosted” VMware Workstation, Microsoft VPC, Parallels
- “Bare metal” VMware ESX, Xen, Microsoft Hyper-V

Virtualization Properties

Isolation of faults and performance

Encapsulation of entire VM state

- Enables snapshots and cloning of VMs

Portability

- Independent of physical hardware
- Enables migration of live, running VMs

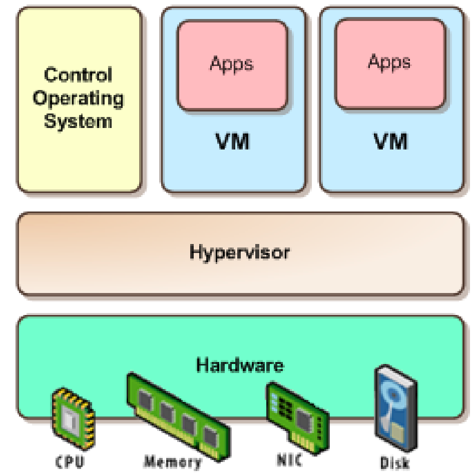
Interposition

- Transformations on instructions, memory, I/O
- Enables transparent resource overcommitment, encryption, compression, replication …

Type 1 vs. Type 2

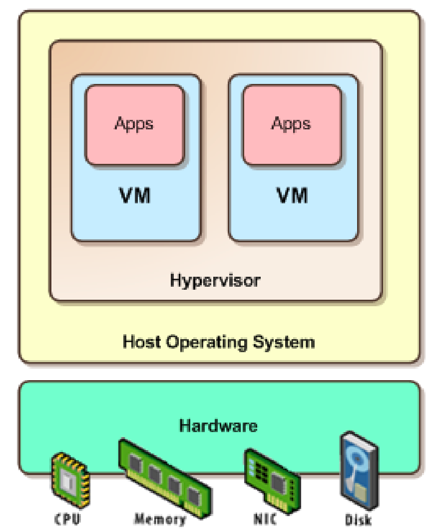
Native/Bare metal (Type 1)

- Higher performance
- ESX, Xen, HyperV



Hosted (Type 2)

- Easier to install
- Leverage host's device drivers
- VMware Workstation, Parallels



Full Virtualization

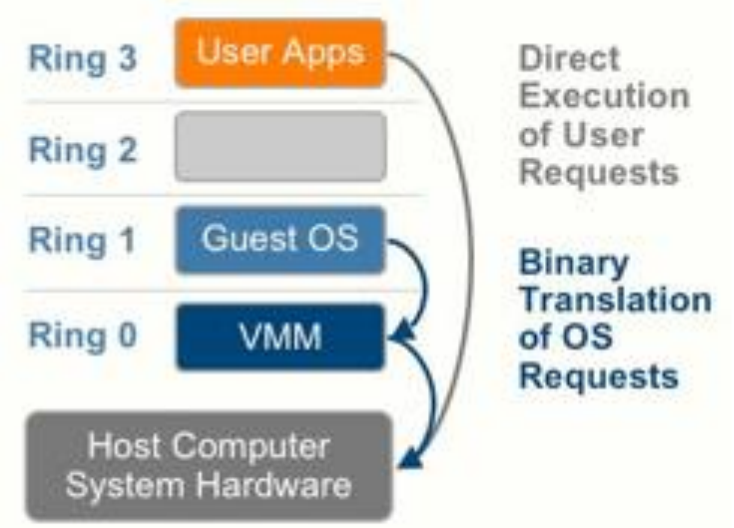
Example: VMware ESX

Functionally identical to underlying physical hardware

Functionality is exposed to the VMs

Allows <u>unmodified</u> guest OS to execute on the VMs

- Transparent to OS: VM looks like the physical machine
- This might result in some performance degradation



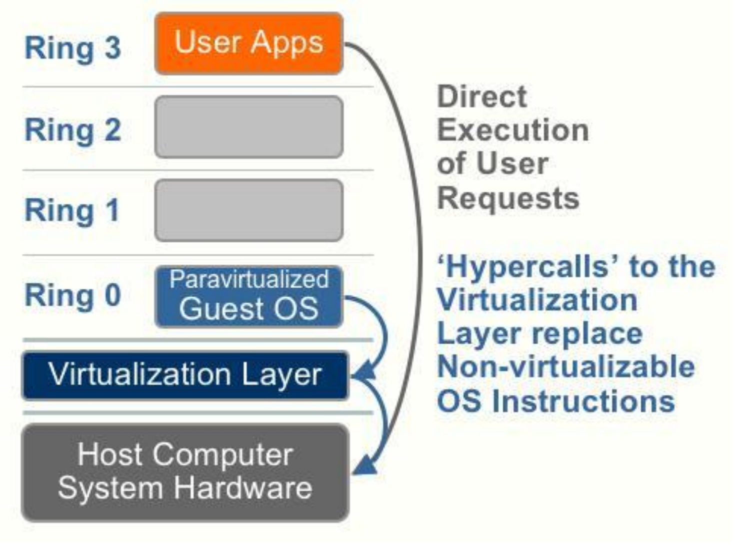
Para-Virtualization

Example: Xen

Virtual hardware abstraction similar, but not identical to the real hardware

Guest OS <u>modified</u> to cooperate with the VMM

- Lower overhead leading to better performance



Example VM Use Cases

- Legacy support (e.g., IBM VM/370 from 1970s)
- Development
- Server consolidation
- Sandboxing / containment
- Cloud computing Infrastructure-as-a-Service

Studying Malware with VMs

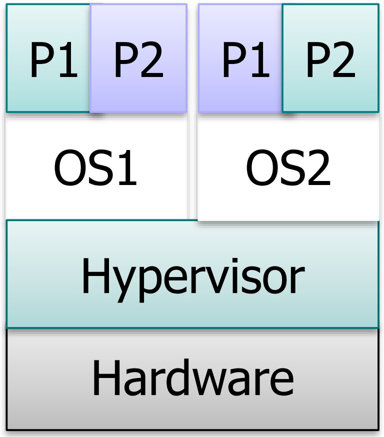
Researchers use VMs to study malware

Example of VM sandboxing

- Hypervisor must confine malicious code

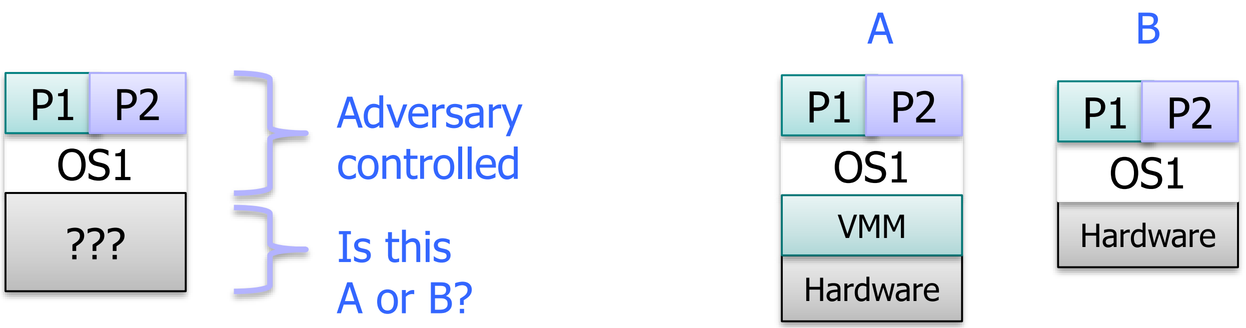
How would you evade analysis as a malware writer?

- Split personalities



VMM Transparency

Garfinkel et al. "Compatibility is Not Transparency: VMM Detection Myths and Reality"



Adversary can detect:

- Para-virtualization
- Logical discrepancies
  - Expected CPU behavior vs virtualized
  - Red pill (Store Interrupt Descriptor Table instr)
- Timing discrepancies
  - Slower use of some resources

Hypervisor Detection

Red Pill Techniques

VM platforms often emulate simple hardware

- VMWare emulates an ancient i440bx chipset

  … but report 8GB RAM, dual CPUs, etc.

Hypervisor introduces time latency variances

- Memory cache behavior differs in presence of hypervisor
- Results in relative time variations for any two operations

Hypervisor shares the TLB with Guest OS

- Guest OS can detect reduced TLB size

… and many more methods [GAWF' 07]

Hypervisor Detection in Browser

Identifying malware web sites: crawl Web, load pages in a browser running in a VM, look for pages that damage VM

Problem: web page can detect it is running in a VM by using timing variations in writing to screen…malware in web page becomes benign when in a VM, evades detection

Hypervisor Security Assumption

Hypervisor 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 hypervisor protect itself and is not buggy

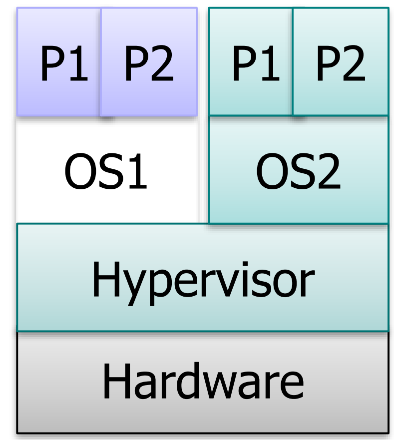
… (some) hypervisors are much simpler than a full OS

Violating Containment

Escape-from-VM

- Vulnerability in VMM or host OS (e.g., Dom0)

Memory management flaws in VMM



Zero-Day Exploit Published for VM Escape Flaw in VirtualBox

by Lucian Constantin on November 8, 2018

A security researcher disclosed a yet unpatched zero-day vulnerability in the popular VirtualBox virtualization software that can be exploited from a guest operating system to break out of the virtual machine and gain access to the host OS.

Violating Isolation

Covert channels between VMs circumvent access controls

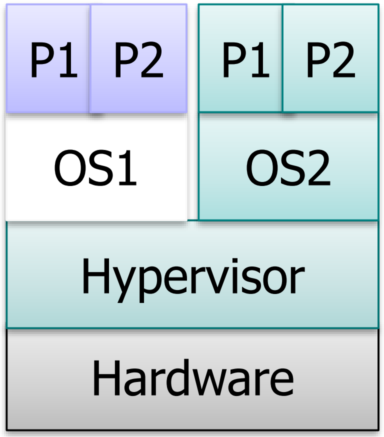
- Bugs in VMM
- Side-effects of resource usage

Degradation-of-service attacks

- Guests might maliciously contend for resources
- Xen scheduler vulnerability

Side channels

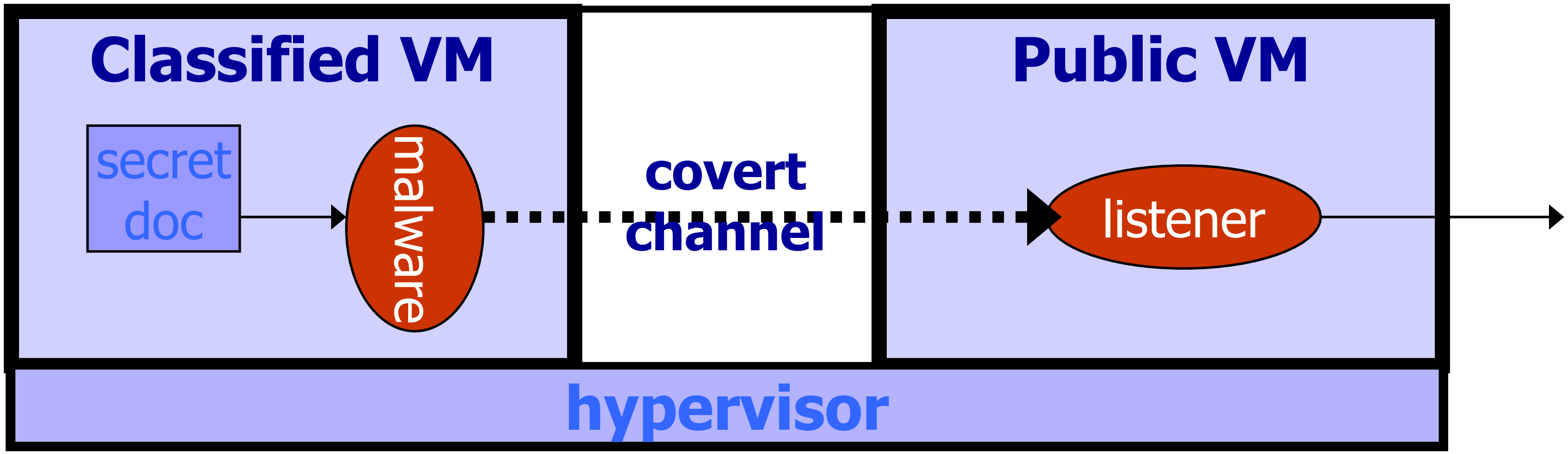
- Spy on other guest via shared resources



Covert Channels

Covert channel: unintended communication channel between isolated components

- Can leak classified data from secure component to public component



An Example Covert Channel

Both VMs use the same underlying hardware

To send a bit $b \in \{0,1\}$ 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

$b = 1 \quad \Rightarrow \quad$ completion-time > threshold

Many covert channels exist in running system:

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