All You Ever Wanted to Know About Virtual Machine Introspection: Introduction

Zhiqiang Lin

Department of Computer Sciences
The University of Texas at Dallas

August 24th, 2015

Outline

Outline

About the Instructor

- Assistant Professor at UT Dallas
- Founding Director of S^3 Lab
- Working with 6 PhDs, 4 MS, 4 Undergraduate
- Research is supported by DARPA, AFOSR, NSF, VMware

About the Instructor

- Assistant Professor at UT Dallas
- Founding Director of S^3 Lab
- Working with 6 PhDs, 4 MS, 4 Undergraduate
- Research is supported by DARPA, AFOSR, NSF, VMware

Goal

- **Building** new systems and automated techniques to secure modern computer systems, including OS kernels and the running software.

About the Instructor

- Assistant Professor at UT Dallas
- Founding Director of S^3 Lab
- Working with 6 PhDs, 4 MS, 4 Undergraduate
- Research is supported by DARPA, AFOSR, NSF, VMware

Goal

- **Building** new systems and automated techniques to secure modern computer systems, including OS kernels and the running software.

Research Interests

- Software security (or binary code analysis)
- Systems security, cloud computing (Virtualization, Introspection)
- Memory or disk data analysis (for introspection, digital forensics, and intrusion detection)

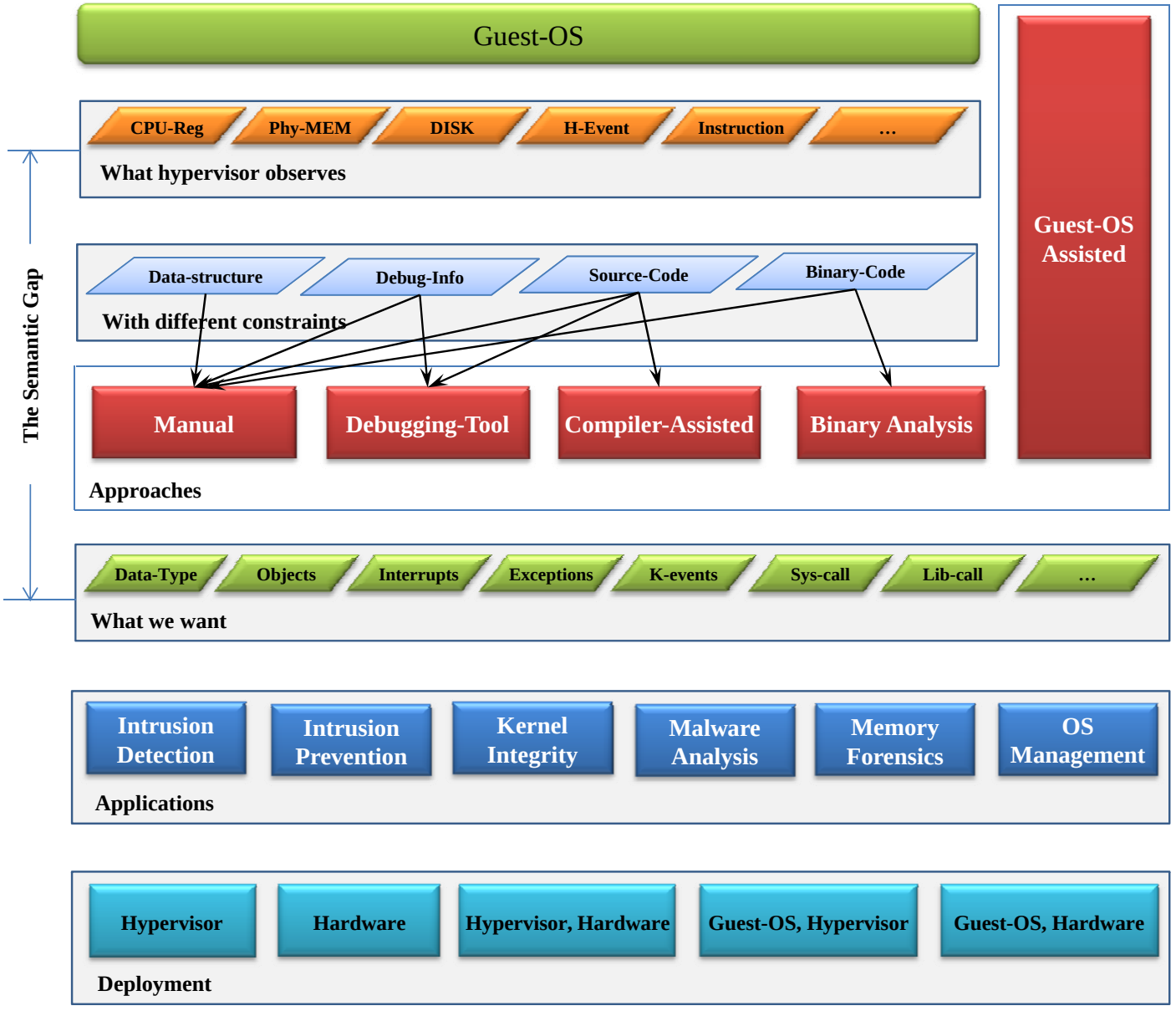
About This Lecture

This lecture aims to provide the students with the necessary knowledge of VMI. It starts from the **basic concept**, to the **principles** behind, and the enabled **applications**.

In particular, based on years of experiences, the instructor will discuss how VMI works essentially, what the challenges are, and how to develop the practical VMI tools.

Hands on labs with pre built VM with the corresponding tool sets will also be provided in this lecture.

The Road Map


Guest-OS
CPU-Reg
Phy-MEM
DISK
H-Event
Instruction
…
What hypervisor observes
Data-structure
Debug-Info
Source-Code
Binary-Code
With different constraints
Guest-OS Assisted
The Semantic Gap
Manual
Debugging-Tool
Compiler-Assisted
Binary Analysis
Approaches
Data-Type
Objects
Interrupts
Exceptions
K-events
Sys-call
Lib-call
…
What we want
Intrusion Detection
Intrusion Prevention
Kernel Integrity
Malware Analysis
Memory Forensics
OS Management
Applications
Hypervisor
Hardware
Hypervisor, Hardware
Guest-OS, Hypervisor
Guest-OS, Hardware
Deployment

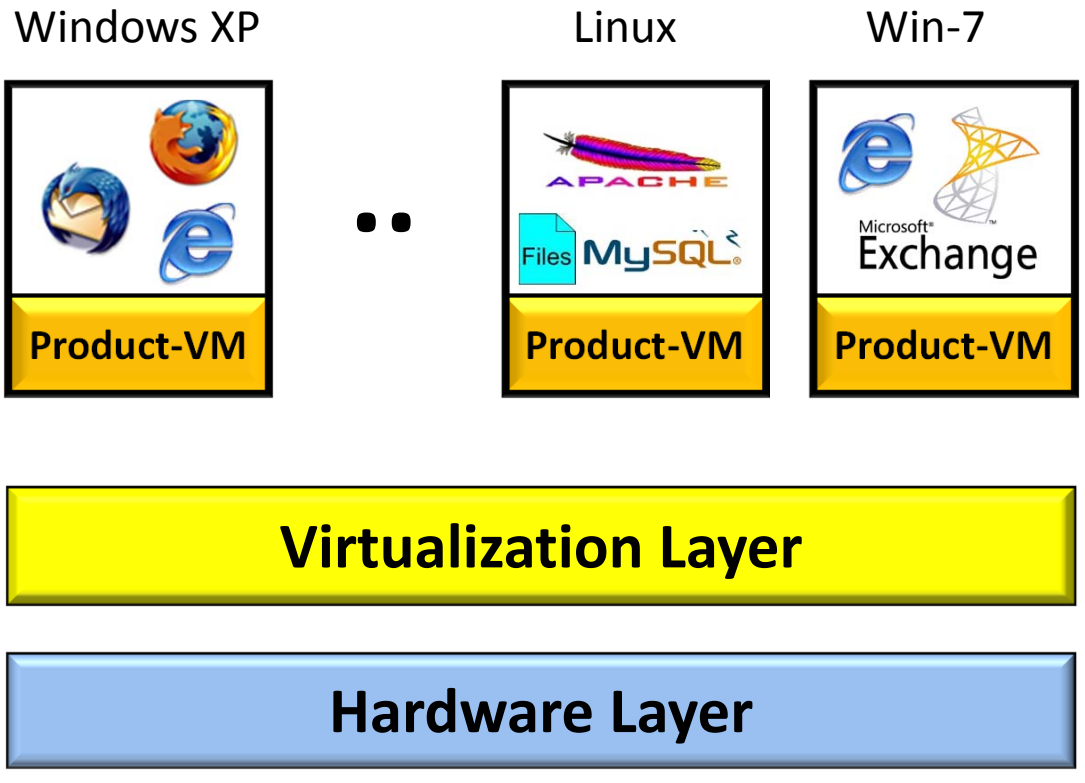
Outline

Today's Cyber

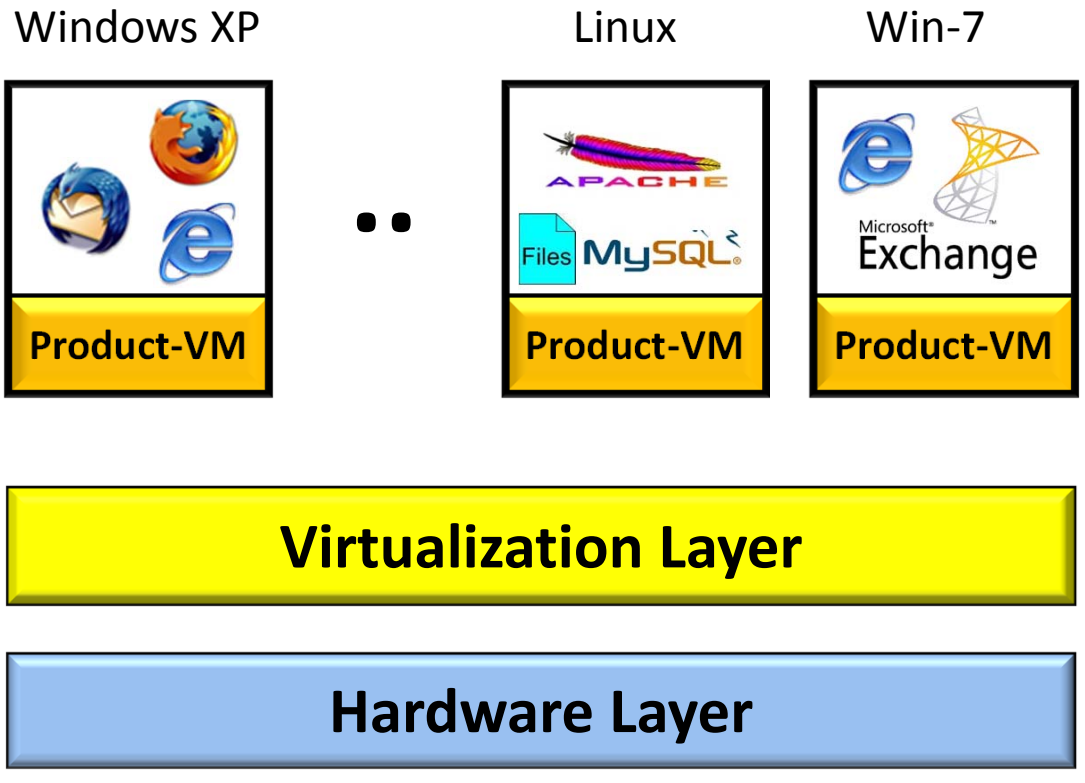
Today's Cyber

http://to-day2.blogspot.com/2013/08/cloud-mobile-social-big-data-monetizing.html

About Virtualization

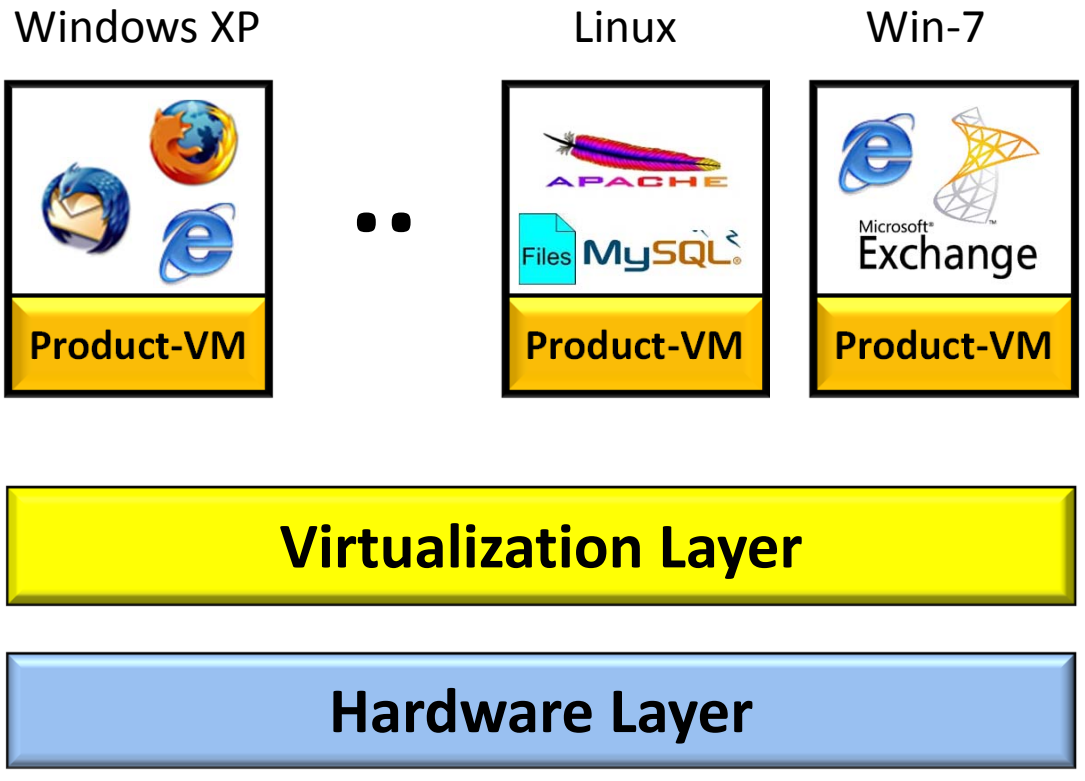


About Virtualization



Virtualization (i.e., hypervisor) [Popek and Goldberg, 1974] has pushed our computing paradigm from **multi-tasking** to **multi-OS**.

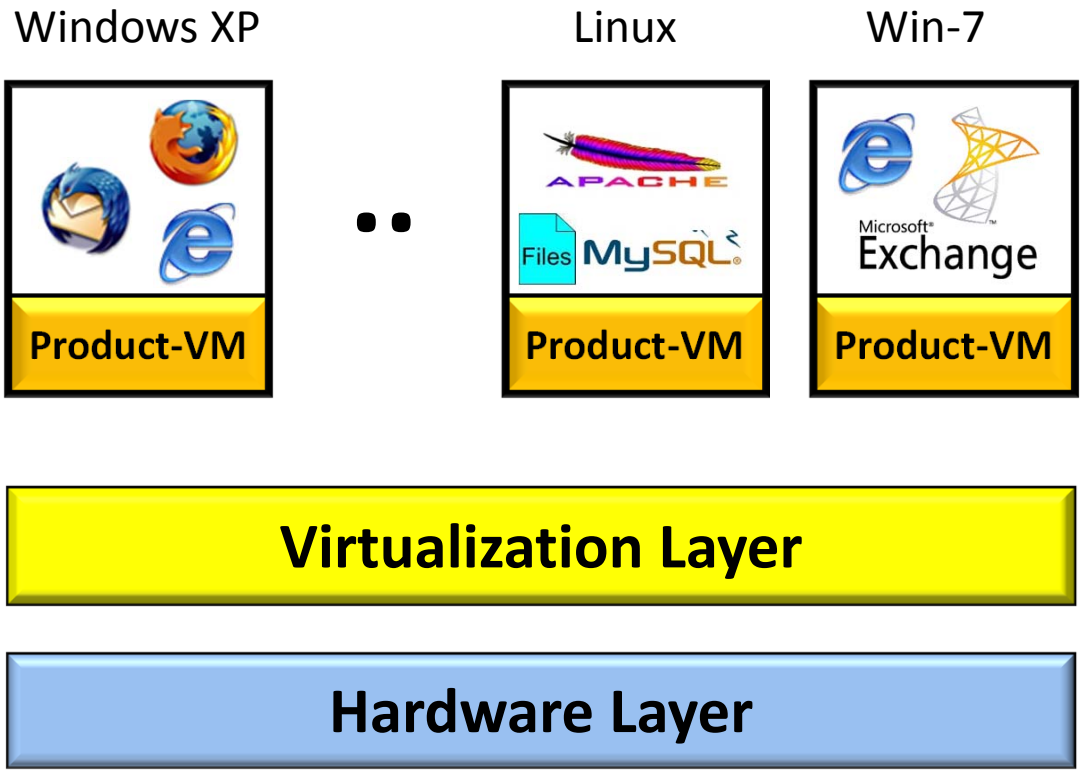
About Virtualization



Virtualization (i.e., hypervisor) [Popek and Goldberg, 1974] has pushed our computing paradigm from **multi-tasking** to **multi-OS**.

Multiplexing, Isolation, Migration, ...

About Virtualization



Virtualization (i.e., hypervisor) [Popek and Goldberg, 1974] has pushed our computing paradigm from **multi-tasking** to **multi-OS**.

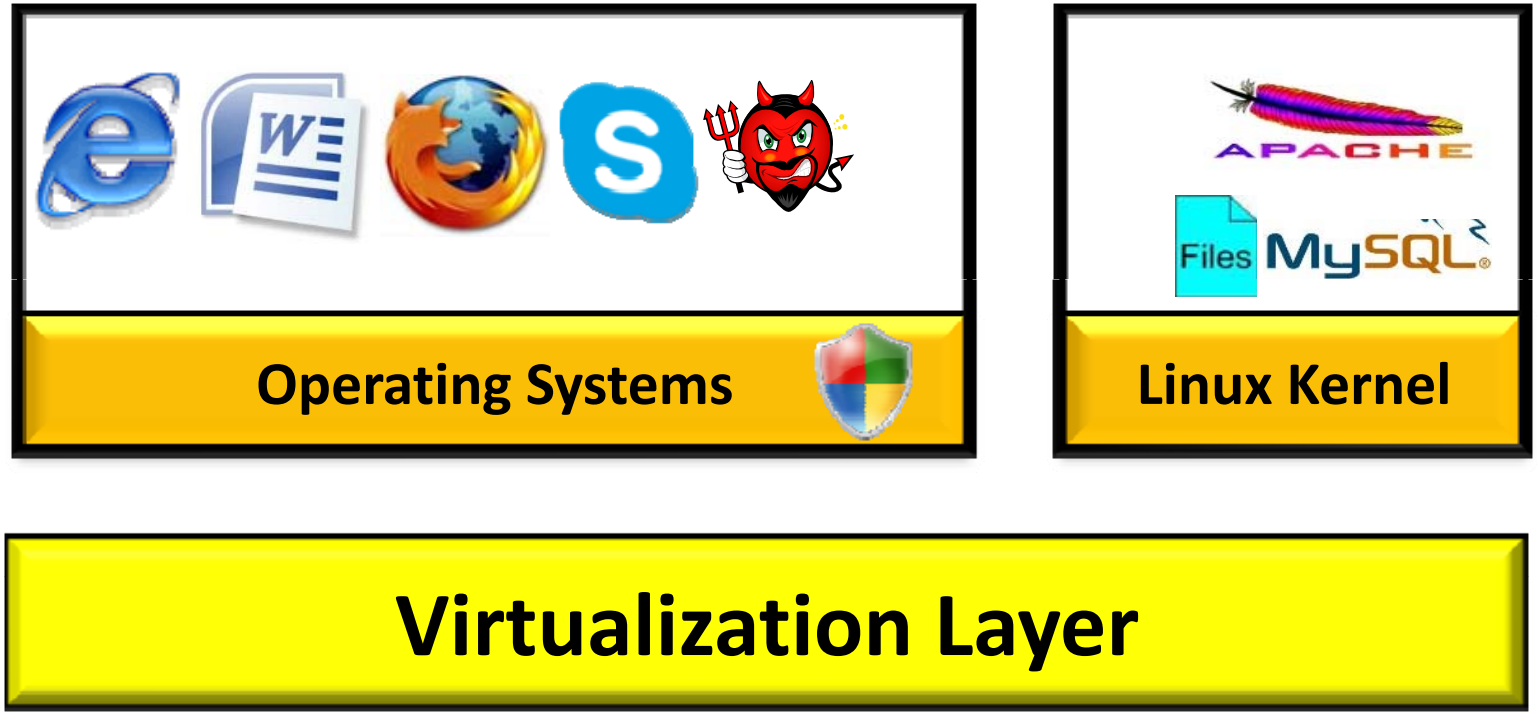
Multiplexing, Isolation, Migration, ...

Xen KVM vmware vSphere ORACLE VirtualBox Microsoft Hyper-V QEMU

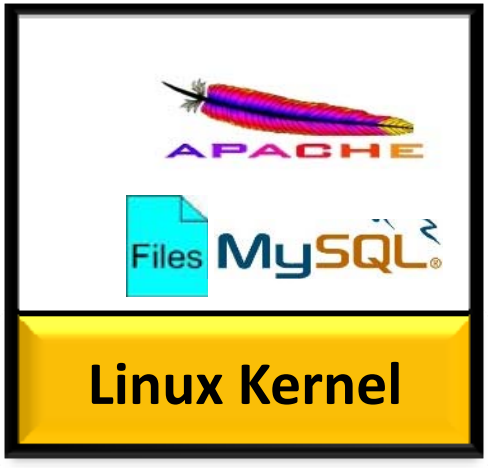
On Anti-Virus Software Evolution

Virtualization Layer

On Anti-Virus Software Evolution

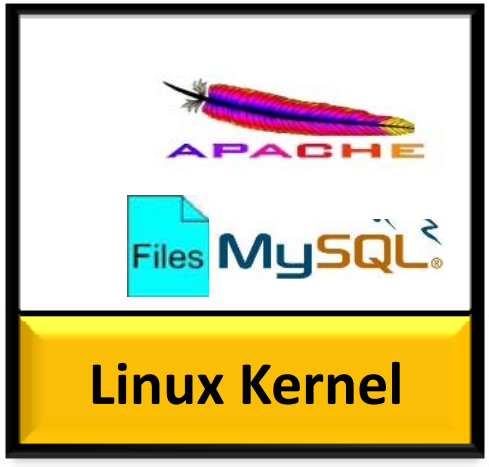


On Anti-Virus Software Evolution

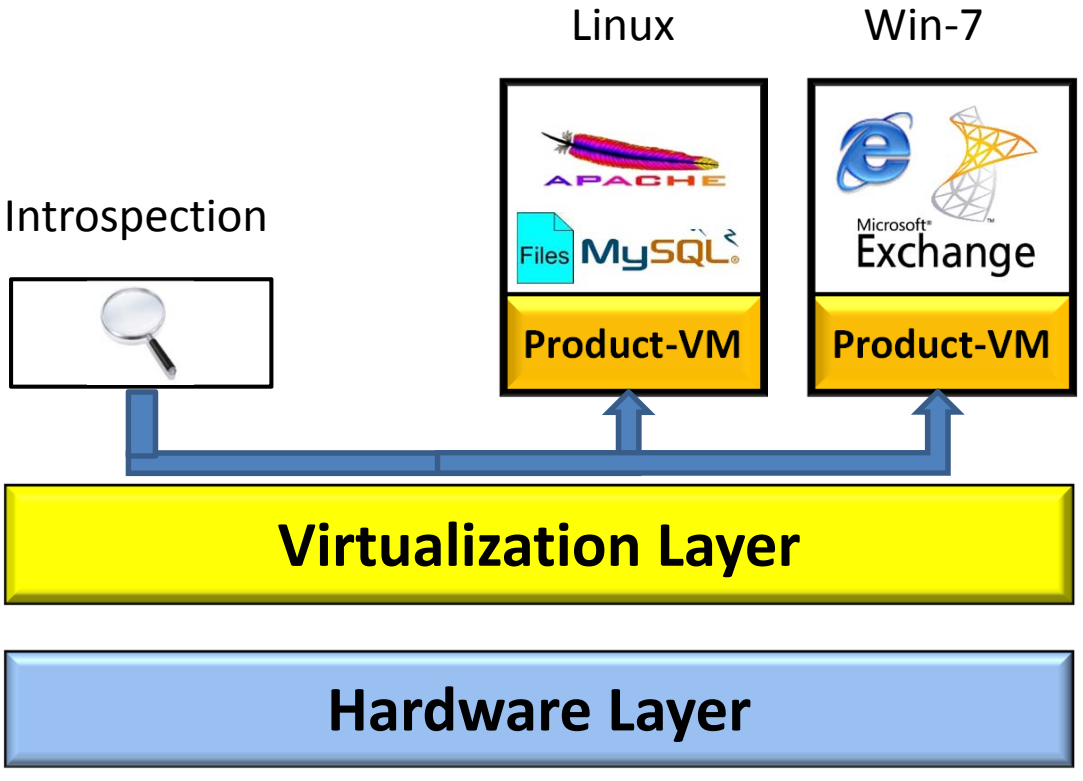


Virtualization Layer

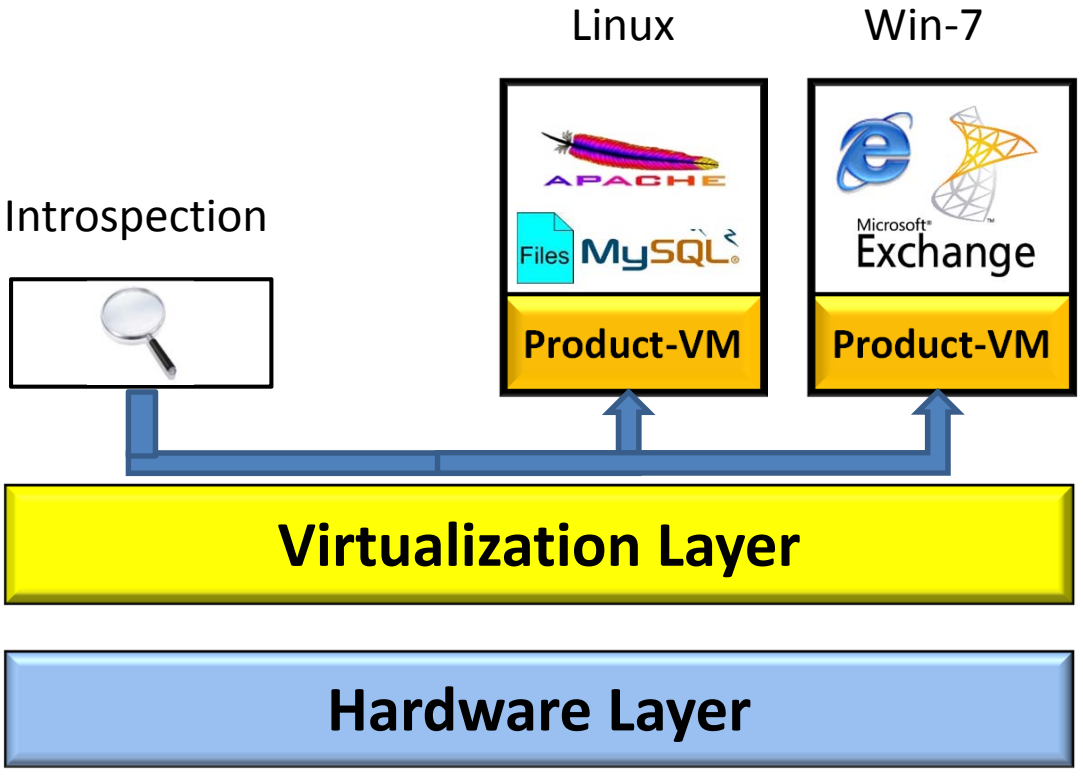
On Anti-Virus Software Evolution



Virtual Machine Introspection (VMI) [Garfinkel et al, NDSS'03]

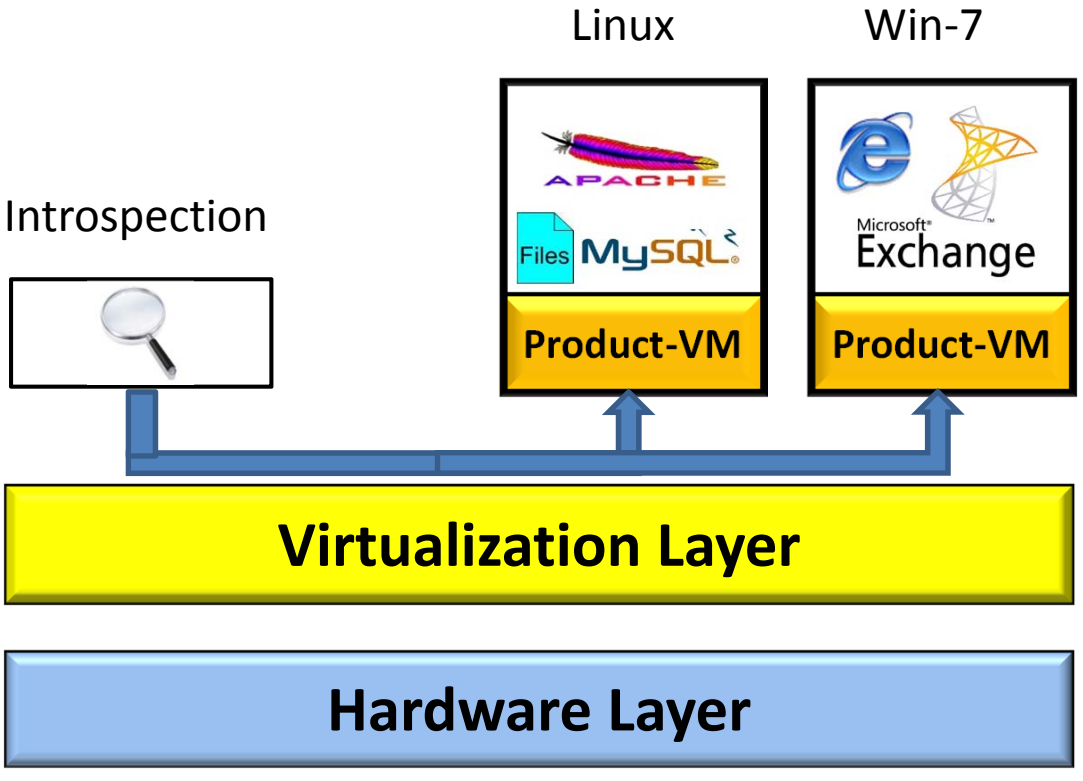


Virtual Machine Introspection (VMI) [Garfinkel et al, NDSS'03]



Using a trusted, dedicated virtualization layer program to monitor the running VMs

Virtual Machine Introspection (VMI) [Garfinkel et al, NDSS'03]



Using a trusted, dedicated virtualization layer program to monitor the running VMs

- Intrusion Detection
- Malware Analysis
- Memory Forensics

Outline

Security monitoring system

$$M(S, P) \rightarrow \{True, False\} \quad (1)$$

where M denotes the security enforcing mechanism, S denotes the current system state, and P denotes the predefined policy.

Security monitoring system

$$M(S, P) \rightarrow \{True, False\} \quad (1)$$

where M denotes the security enforcing mechanism, S denotes the current system state, and P denotes the predefined policy.

- If the current state S satisfies the security policy P, then it is in a secure state (*True*)
- If M is an online mechanism, it can allow continued execution; Otherwise, snapshot based approach (e.g., for forensics).

In-VM Inspection

In-VM Inspection

Advantages

- **Rich Abstractions**. Plenty of abstractions for in-VM monitors to extract the OS and process state.
- **Fast Speed**. In-VM state can be directly accessed, and in-VM enforcement can be instantly executed without any trapping into hypervisor.

In-VM Inspection

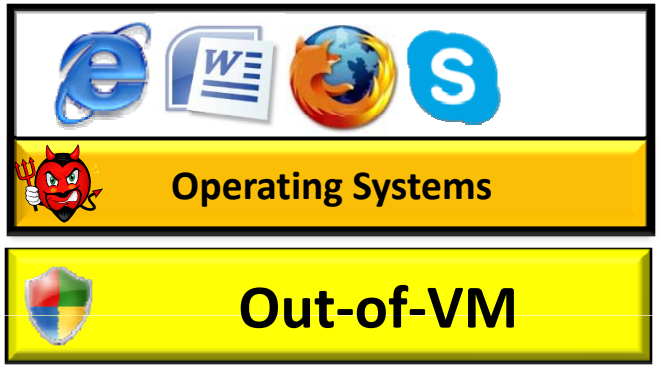
Advantages

- **Rich Abstractions**. Plenty of abstractions for in-VM monitors to extract the OS and process state.
- **Fast Speed**. In-VM state can be directly accessed, and in-VM enforcement can be instantly executed without any trapping into hypervisor.

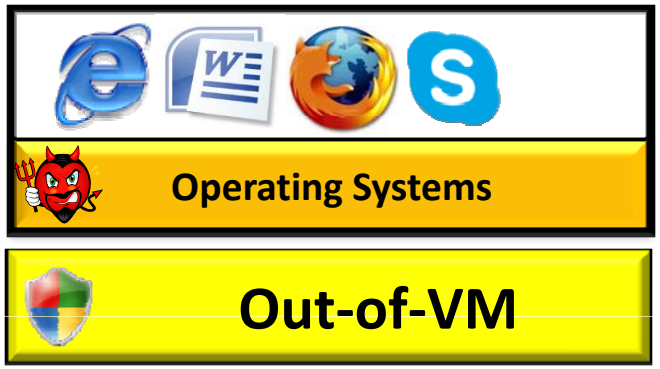
Disadvantages

- The security monitoring system can be disabled
- False State can be generated to mislead the detection. Log files, proc files
- The Security Policy and The Security Enforcing Mechanism can be tampered

Out-of-VM Introspection



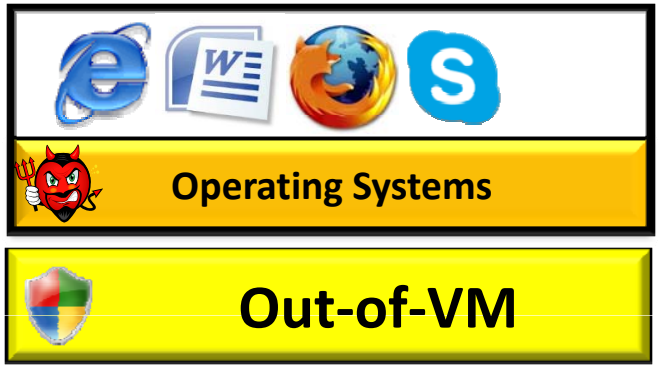
Out-of-VM Introspection



Advantages

- Strong Isolation (Tamper Resilient)
- Transparent Deployment
- Complete View
- High Cost-Saving
- Less Vulnerability

Out-of-VM Introspection



Advantages

- Strong Isolation (Tamper Resilient)
- Transparent Deployment
- Complete View
- High Cost-Saving
- Less Vulnerability

Disadvantages

- **No Abstractions.** No guest OS abstractions, no system calls, no file descriptor, no variables.
- **Slow Speed.** Additional address translation, and world switching.

Outline

Hypervisor Definition: Bare metal

Type 1 (**bare metal**) hypervisors, which run directly on the host's hardware to control the hardware and monitor the guest OS. Typical examples of such hypervisors include Xen, VMware ESX, and Microsoft Hyper-V.

Hypervisor Definition: Hosted

Type 2 (**hosted**) hypervisors, which run within a traditional OS. That is, a hosted hypervisor adds a distinct software layer atop the host OS, and the guest OS becomes a third software layer above the hardware. Well-known examples include KVM/VMware Workstation/VirtualBox/QEMU.

Hypervisor Definition: Native

Native hypervisors that directly push the guest code to execute natively on the hardware with hardware virtualization.

Hypervisor Definition: Emulation

Emulation hypervisors that translate each guest instruction for an emulated execution with software virtualization.