

Requirements for an Integrity-Protected Hypervisor on the x86 Hardware Virtualized Architecture

Amit Vasudevan, Jonathan M. McCune, Ning Qu, Leendert van Doorn, Adrian Perrig

CyLab, Carnegie Mellon University

June 22, 2010

Outline

- Introduction
- Elements of x86 Hardware Virtualization
- Integrity Protected Hypervisor
- Guest and Hardware Requirements
- Popular Hypervisors
- Conclusions

Introduction/Motivation

- Virtualization => VMM a.k.a Hypervisor
- Hypervisor = maximally privileged software component from guest's TCB perspective
- Today's Hypervisors have their share of vulnerabilities
- Data secrecy and availability can be guaranteed only if there is integrity
- Are today's virtualization and security extensions on the x86 platform sufficient to maintain hypervisor integrity?

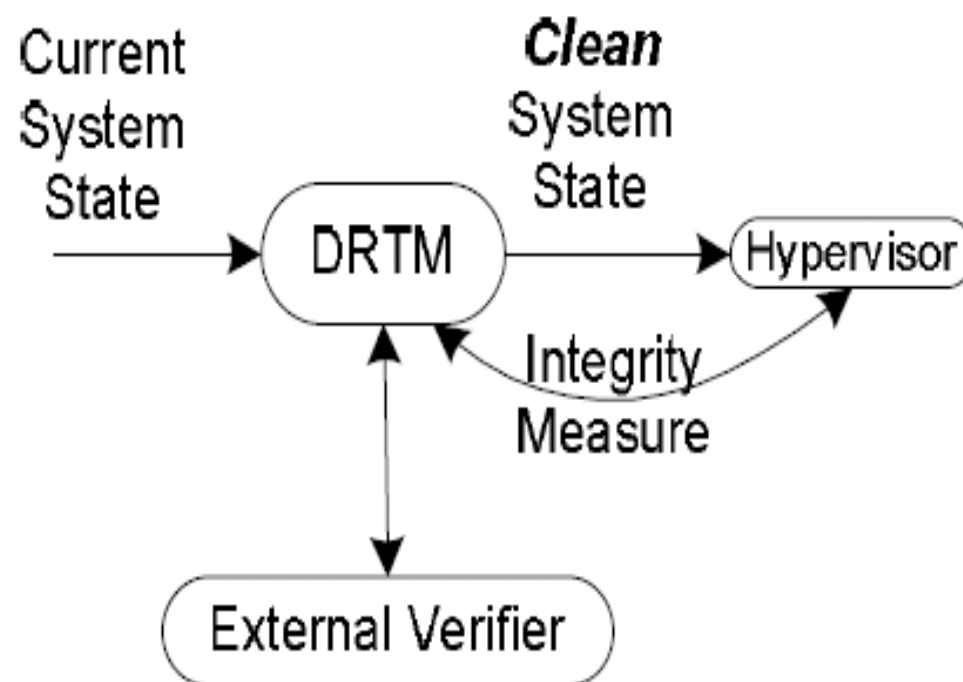
Elements of x86 Hardware Virtualization

- Only system components that can directly access memory or mediate memory accesses become critical to preserving hypervisor integrity
- Hardware Elements
 - CPU (SMM, MMU, Microcode)
 - Northbridge (IOMMU) and Southbridge
- Software Elements
 - BIOS/UEFI, Option ROMs, ACPI Scripts and Other code

Integrity Protected Hypervisor

- Assumption: The target system on which an integrity-protected hypervisor runs is physically protected.
- The “12” Commandments
 - Startup Rules
 - Runtime Rules
 - Hypervisor Design Rule

Startup Rules



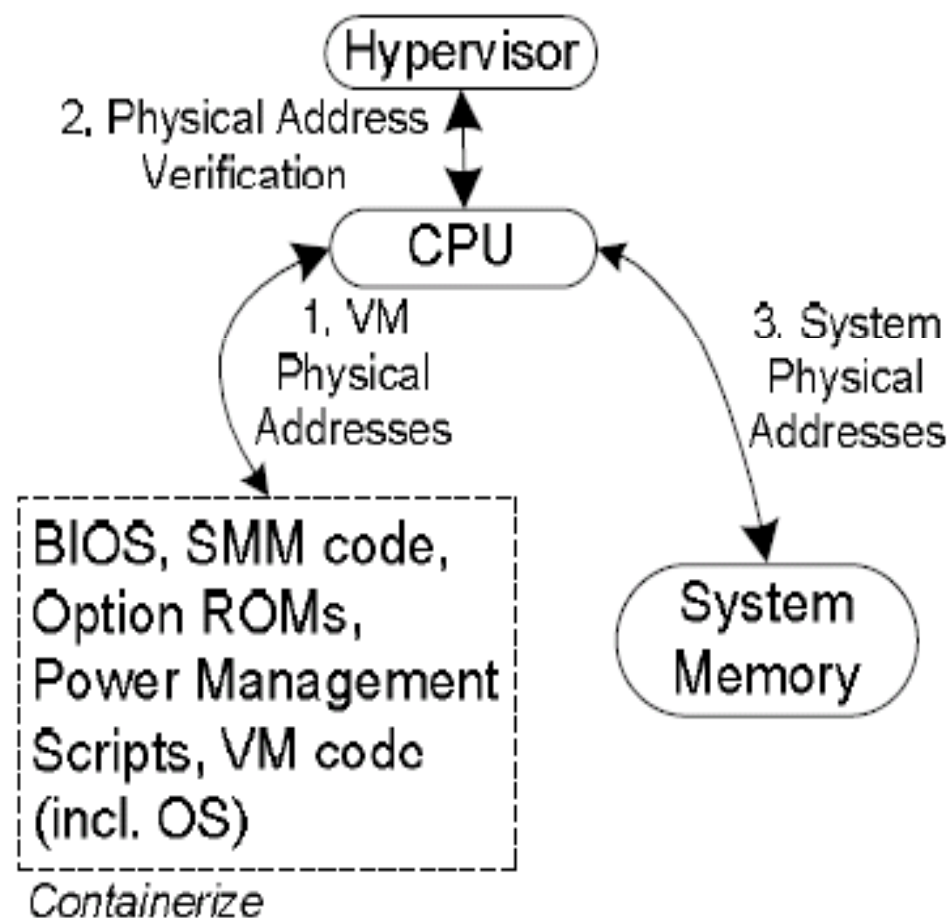
Startup Rules

- Rule-1: An Integrity Protected Hypervisor must be initialized via the creation of a Dynamic Root of Trust.
 - Legacy code that cannot be trusted
 - Malicious option ROMs
 - DRT provides initialization without breaking legacy compatibility

Startup Rules

- Rule-2: A dynamic root of trust mechanism must allow for an external verifier to ascertain the identity (e.g., cryptographic hash) of the memory region (code and data) in the new execution environment.
 - DRT only guarantees clean launch environment, hypervisor integrity can already be compromised
 - We need to verify identity of the loaded hypervisor

Runtime Rules



Runtime Rules

- Rule-3: An integrity-protected hypervisor must employ physical memory virtualization to prevent any code executing within a VM from accessing hypervisor memory regions.
 - Code running within a VM may manipulate the MMU's virtual memory data structures to map and access hypervisor memory regions

Runtime Rules

- Rule-4: An integrity-protected hypervisor must execute its core in the highest privilege level so it can interpose on critical system operations.
 - Hypervisor uses privileged instructions
 - Should be able to intercept critical system operations. Eg. Device I/O, CPU control registers etc.

Runtime Rules

- Rule-5: An integrity-protected hypervisor must have an independent set of critical CPU registers and must sanitize values of CPU data registers during control transfers to and from VMs.
 - Sharing critical CPU registers (MSRs/MTRRs)
 - CPU Data registers

Runtime Rules

- Rule-6: An integrity-protected hypervisor requires the MMU to maintain independent states for the hypervisor and guest environments.
 - MMU Registers
 - Translation Lookaside Buffers (TLB)

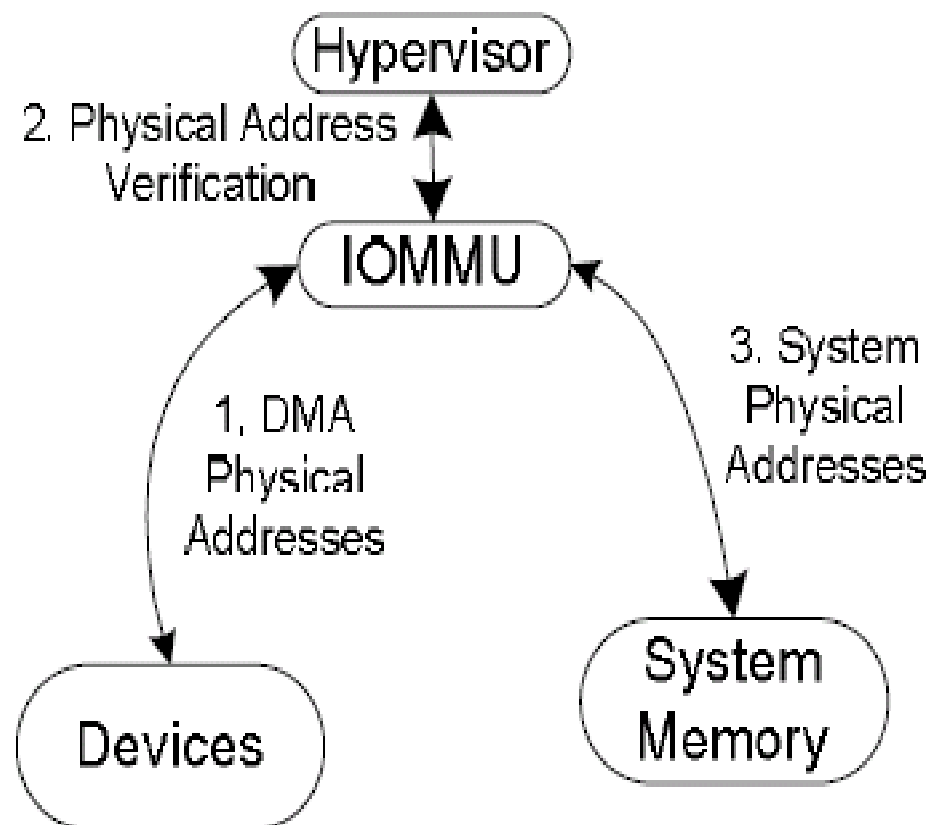
Runtime Rules

- Rule-7: An integrity-protected hypervisor must intercept all x86 hardware virtualization instructions.
 - Virtualization specific instructions are used to create, modify, run and terminate guest environments and to save and load guest environment states
 - Guest must never be able to execute these instructions natively

Runtime Rules

- Rule-8: An integrity-protected hypervisor must containerize any System Management Mode code and BIOS, option ROM or Power Management Scripts it uses.
 - A buggy or malicious #SMI handler can access memory regions belonging to the hypervisor and compromise its integrity
 - Malicious code can be embedded within the BIOS, option ROM, or Power Management Scripts and these in turn can alter hypervisor memory regions

Runtime Rules



Runtime Rules

- Rule-9: An integrity-protected hypervisor must prevent system devices from directly accessing hypervisor memory.
 - USB, Firewire, Storage and Network devices can directly access physical memory via DMA, potentially bypassing the hypervisor.
 - Malicious firmware on a device can also accomplish the same goal by replacing legitimate physical memory addresses passed to it with hypervisor physical memory regions.

Runtime Rules

Rule-10: An integrity-protected hypervisor must enumerate all system devices at startup and be able to detect hot-plug devices at runtime

- Restrict devices from accessing hypervisor memory regions → configure memory access restrictions for every device within the system → need to uniquely identify each device
- Device configuration can change → hypervisor must scan on startup as well as during runtime

Runtime Rules

Rule-11: An integrity-protected hypervisor must prevent access to critical system devices at all times.

- Northbridge, Southbridge and IOMMU must be protected at all times.

Design Rule

Rule-12: An integrity-protected hypervisors' code must be free of vulnerabilities..

- Hypervisor configuration and runtime interfaces pose significant risk to hypervisor integrity
- Operating logic should be simple and code-base must be within limits to perform manual and analytical audits.

Guest and Hardware Requirements

- Multiple Guests
 - Guest BIOS Calls
 - Sharing between guests
- Hardware Considerations
- *Experience with devices*
 - South Bridge Renders DRTM / PCR 17 Unusable
 - SMRAM locked by the BIOS resulting in no SMI intercept
 - ITPM
- Importance of Correct hardware

Popular Hypervisors

Integrity Rules	Hypervisors				
	VMware	Xen	Hyper-V	L4	SecVisor
1. An integrity-protected hypervisor must be initialized via the creation of a dynamic root of trust.	✗	✓	✗	✗	✓
2. A dynamic root of trust mechanism must allow for an external verifier to ascertain the identity of the code that has received control in the new execution environment.	✓	✓	✓	✓	✓
3. An integrity-protected hypervisor must employ physical memory virtualization to prevent any code executing within a VM from accessing hypervisor memory regions.	✓	✓	✓	✓	✓
4. An integrity-protected hypervisor must execute its core in the highest privilege level that allows it to interpose on critical system operations.	✓	✓	✓	✓	✓
5. An integrity-protected hypervisor must have an independent set of critical CPU registers and must sanitize values of CPU data registers during control transfers to and from VMs.	✓	✓	✓	✓	✓
6. An integrity-protected hypervisor requires the MMU to maintain independent states for the hypervisor and guest environments.	✓	✓	✓	✓	✓
7. An integrity-protected hypervisor must intercept all x86 hardware virtualization instructions.	✓	✓	✓	✓	✓
8. An integrity-protected hypervisor must containerize any SMM code, BIOS, option ROM or Power Management Script it uses.	✗	✗	✗	✗	✗
9. An integrity-protected hypervisor must prevent system devices from directly accessing hypervisor memory regions.	✓	✓	✓	✓	✓
10. An integrity-protected hypervisor must enumerate all system devices at startup and be able to detect hot-plug devices at runtime.	✓	✓	✓	✓	✓
11. An integrity-protected hypervisor must prevent access to critical system devices at all times.	✓	✓	✓	✓	✓
12. An integrity-protected hypervisors' code must be free of vulnerabilities.	✗	✗	✗	✗	✓*

Conclusions

- We explored the low-level details of x86 hardware virtualization and established rules that an integrity-protected hypervisor must follow.
- In theory the latest x86 hardware contains sufficient support to integrity-protect a hypervisor, in practice an integrity-protected hypervisor cannot be realized on today's x86 hardware platforms
 - SMM mode
- Unable to support arbitrarily many legacy guests
 - Sharing devices and data between multiple guests coupled with guest BIOS invocations significantly complicates the hypervisor
- Hardware must be selected with great care

Thank you!
amitvasudevan@acm.org