Intel x86 Hardware Security Primitives
“What tools are in the tool-chest?”

Mr. Jacob Torrey

March 5, 2015
Dartmouth College

@JacobTorrey
torreyj@ainfosec.com
Linkedin.com/in/jacobtorrey

153 Brooks Road, Rome, NY | 315.336.3306 | http://ainfosec.com

Introduction

- In efforts to ease system consolidation and optimize resource use, Intel (and ARM, AMD…) have provided hardware extensions to support hardware-backed security and virtualization of disparate OSes on a single physical machine
- This lecture aims to provide a short introduction to these extensions and how to utilize them in future research (there is a lot of emergent/weird machine behavior here)
- Slides will be provided for reference, please don't hesitate to jump in with questions at any time

Full vs. Paravirtualization

- **In this talk, focus will mostly be on full virtualization**
  - Guest OS is unmodified, generally thinks it is running without VMM
- **Paravirtualization is where modifications to the guest OS are made to simplify**
  - For example: disk driver talks to VMM directly, rather than trapping on MMIO/PIO requests
  - PV drivers still used on fully virtualized VMMs for speed and management advantages
- **Intel VT-x enables full virtualization**
  - From architecture side, more interesting (to me at least)

Outline

- **Gap Analysis – From OS to VMM**
- **Technical Overviews**
  - General architecture
  - Memory management/isolation
  - VMCS mechanics
  - Exit conditions
  - AES-NI
  - Asides: TXT & SMM/STM
- **Interesting Research**
  - Recovering private keys from VM side-channels
  - MoRE – VT-x + TLB splitting
  - TRESOR
- **Concluding Remarks**
- **Questions**

Gap Analysis

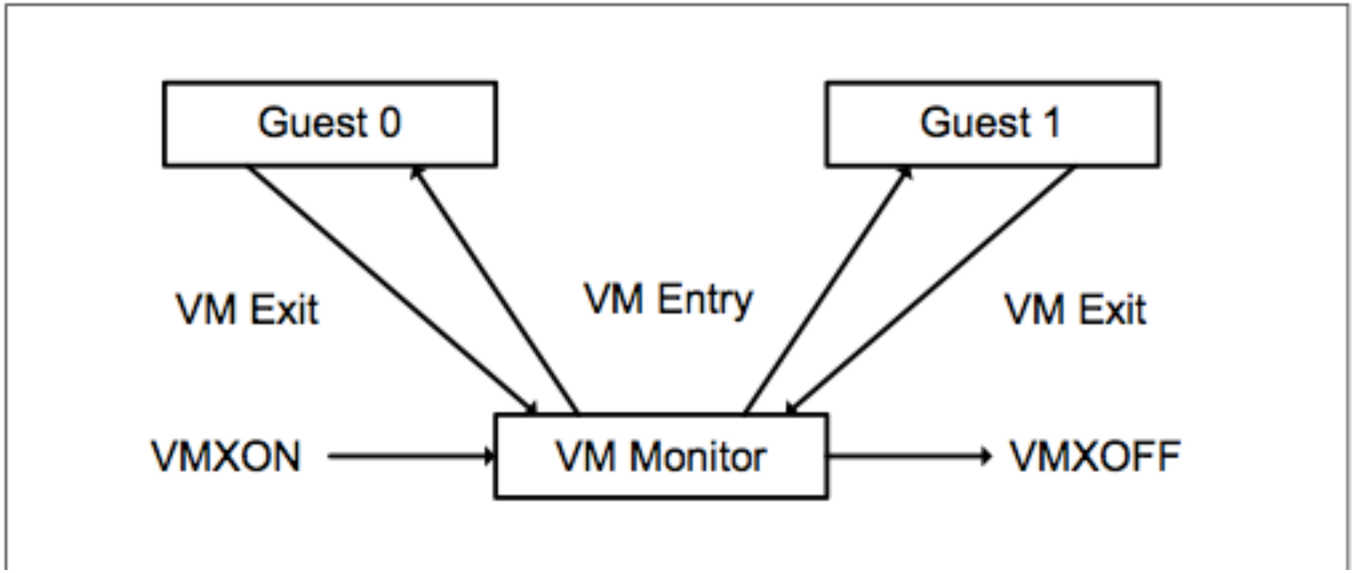
“Ring 0 to Ring -1”

- **In short, a virtual machine monitor (VMM aka hypervisor) is a kernel where OSes are ‘applications’**
  - Can abstract memory (guest physical addresses to machine physical)
  - Multiplex between OSes and trap on specified exceptions/ violations
  - Provide a consistent & abstracted view of hardware

- **Well-designed architecture (mostly) cleans a lot of cruft needed for legacy support**
  - Originally no support for real-mode
  - 64-bit aware, no PAE needed
  - Allows VMM to be very small code base

VT-x in a Figure

- Shows VMM ‘slot’, and the process for transitioning to and from from multiple guests



- From Intel SDM 3C – Official VT-x specification document

The Growth of VMMs

- **Many single OS systems now are VMs**
- **Hyper-V comes by default installed on Win 8+**
- **Linux KVM is merged with kernel mainline**
- **The line between guest, host and VMM is increasingly blurred**
  - Host OS is more privileged than VMM userspace
  - Host OS is less privileged than VMM root

Technical Overview

General Architecture

- **Separated into VMX root and non-root mode**
  - VMXON, VMXOFF and VM Exits/Enters switch between two modes
  - Can be initiated from either Ring 0 or SMM (“Ring -2”)
- **VMM sets up task_struct-like VM control structure (VMCS) for each VM**
  - Specifies what events will trigger VM Exit
- **Some events always trigger VM Exit**
  - CPUID, RDTSC, etc…
  - Can be used to determine if a OS is being maliciously virtualized

Technical Overview

General Architecture II

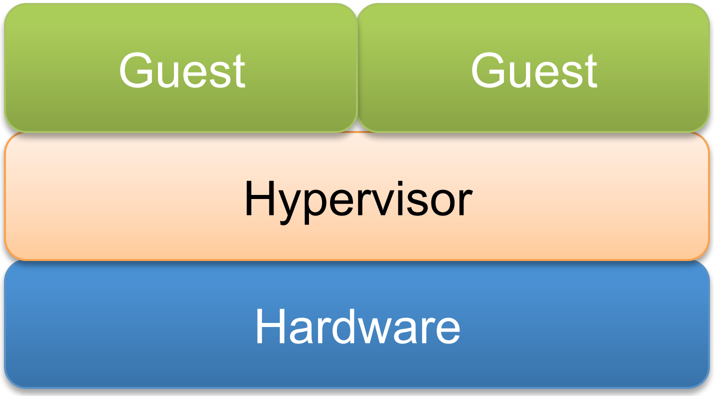
- **VMM is protected from rogue guests, and guests benefit from some protections from each other**
  - Performance and cost were driving factors in implementation

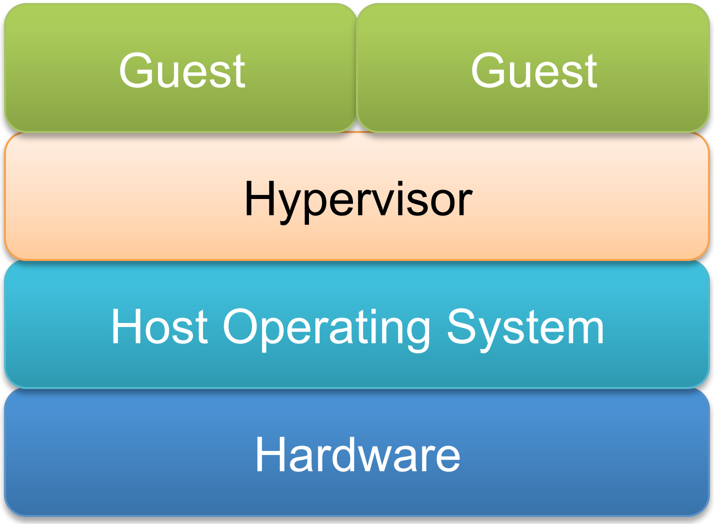
- **Couples with other Intel technologies for greater assurances**
  - VT-d: Prevents hardware devices from DMAing memory to arbitrary memory
  - TXT: Allows a measured launch of a hypervisor at any point and creates a dynamic root-of-trust
  - EPT/VPID: Allows hardware to take a bigger role in memory and cache separation and management

Technical Overview

VMM Architecture

- **VMMs are either Type-I or Type-II**





- **Examples of Type-I**
  - Xen, VMWare ESX, Hyper-V
- **Examples of Type-II**
  - VirtualBox, KVM, VMWare Player

Technical Overview

VMM Architecture II

- **Different types lead to different hardware multiplexing models**

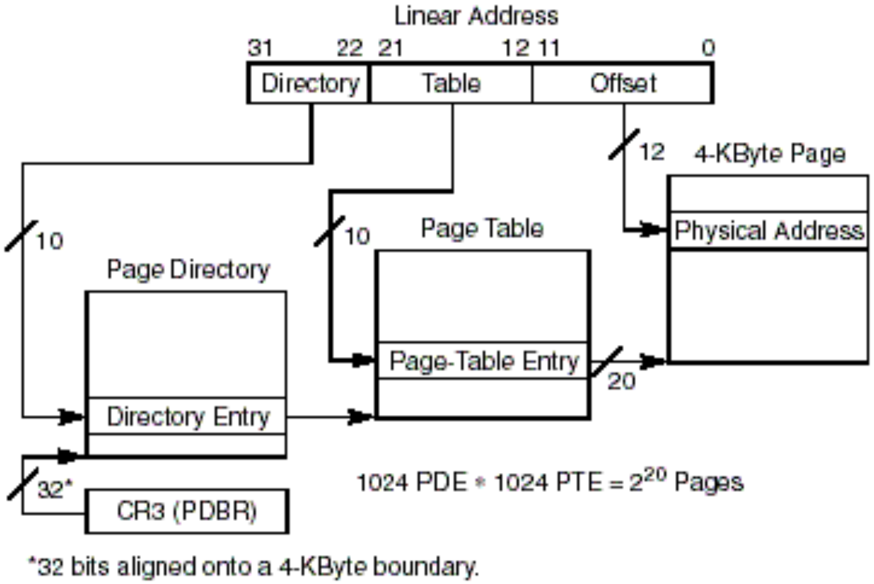
- **Type-I VMMs generally have a control domain (dom0) which can directly talk to hardware and multiplex all requests from guests – don't want to need drivers in VMM**
  - More secure and isolated, no full OS in TCB

- **Type-II VMMs use the hardware drivers of host OS**
  - Simpler to install, just an application on host OS

Technical Background

Virtual Memory

- **Paging provides an operating system with a means to organize physical memory while at the same time, providing executables with an abstracted, contiguous view of memory**

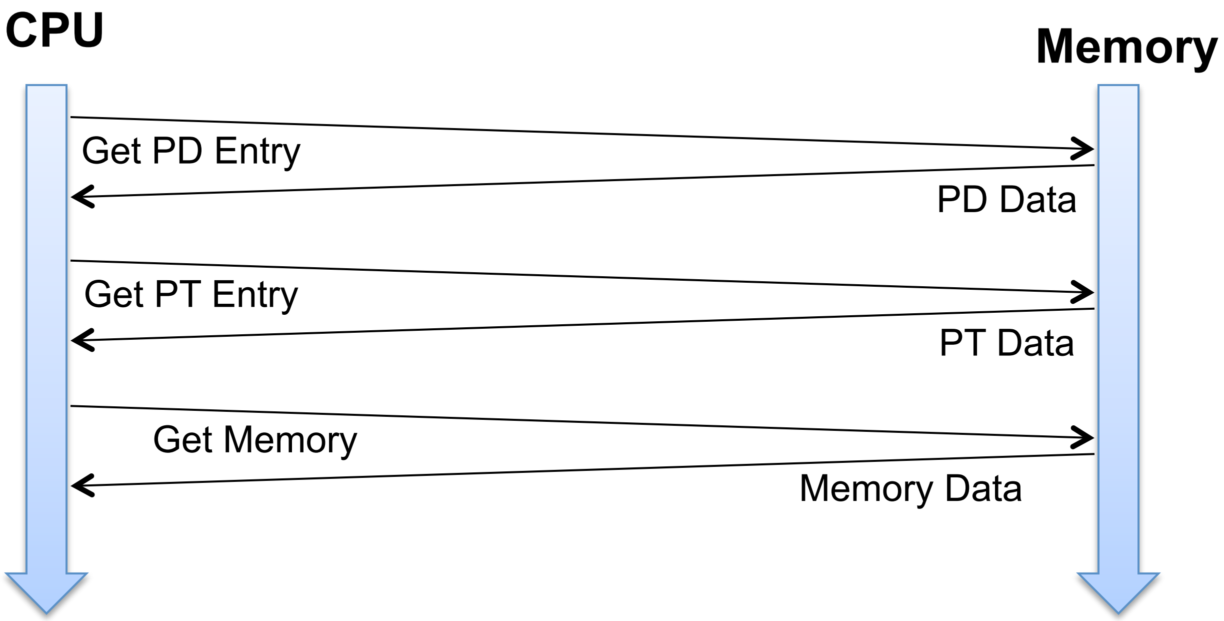


Viralpatal.net

Technical Overview

Page Translation

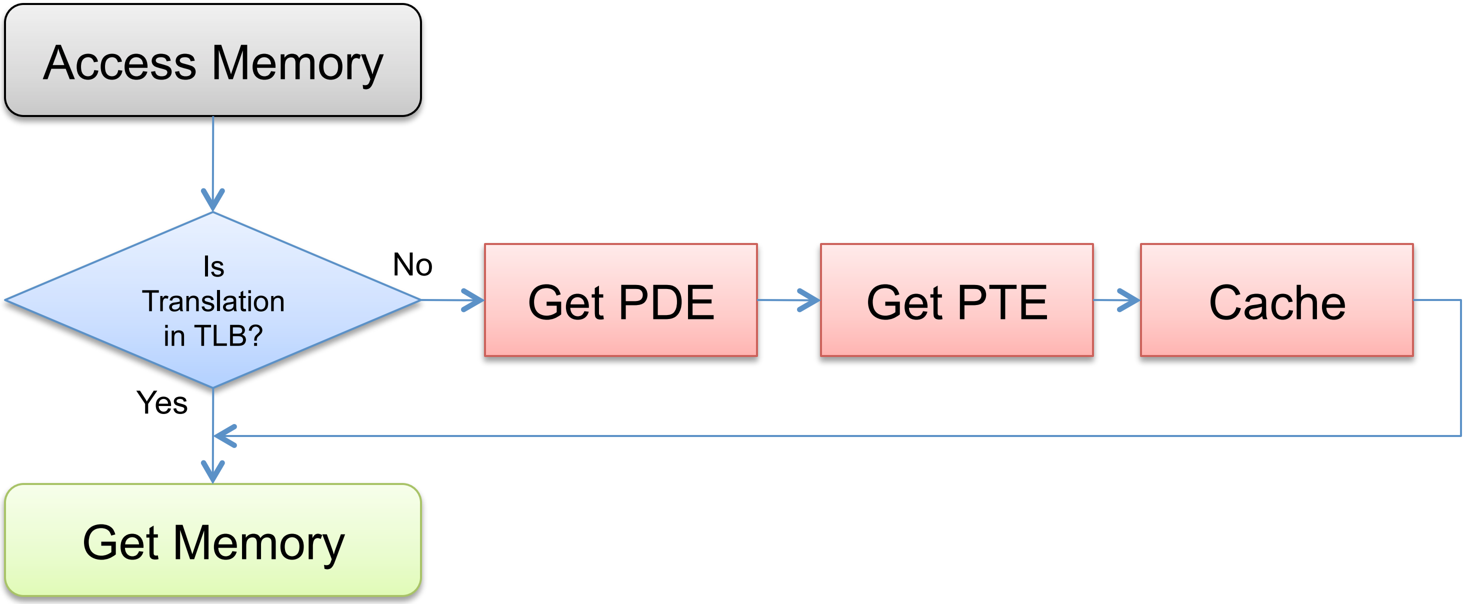
- **Every memory access requires several memory bus transactions to perform page translation**
  - This is slow!



Technical Overview

Translation Lookaside Buffer

- **The solution to this problem is to cache previous translations in a buffer called the Translation Lookaside Buffer (TLB)**



Technical Overview

Memory Management

- **Extended page tables (EPT) adds additional levels to traditional virtual memory hierarchy**
  - Maps "guest physical" to "machine physical"
  - Triggers EPT Fault VM Exit instead of page fault
  - Allows OS to manage memory without VMM interference
  - Implements new instructions similar to INVLPG

- **VM process ID (VPID) adds a word to each TLB line with the VM ID (VMM = ID 0) to prevent performance hit from VM Exit TLB flush**

Technical Overview

VMCS Mechanics

- Think of a VMCS as a task state segment (TSS) or task_struct for OS VMs
- One VMCS PTR per processor, points to currently active VMCS
- VMCS stores guest and host state, exit conditions and pointers to other related structures
- Not directly accessible to memory reads/writes, requires a specialized instruction to access (VMREAD/VMWRITE)

Technical Overview

VM Exit Conditions

- **Among others, the VMCS can be configured to trap on:**
  - Interrupts
  - Memory faults (akin to page faults)
  - IO access (port IO)
  - Certain privileged instructions
    - MOV to control registers
    - RDMSR/WRMSR
    - RDRAND
    - Etc…

- **When trapping to VMM, provides all guest registers/ state and exit condition**

AES-NI

- In response to software-based caching attacks on AES, Intel released instruction set to support AES
- Hardware logic is faster, and more protected
- Supports 128-bit and 256-bit AES
- Provides primitives, still requires engineering to make a safe system on top of these

Technical Aside

TXT & SMM

- **Intel trusted execution technology (TXT) provides the ability to establish a dynamic root-of-trust**
  - Sets TPM (hardware crypto co-processor) into special mode as well as CPU(s) and launches measured launch environment
  - Removes legacy BIOS and additional untrusted software from trusted computing base (TCB)
- **Intel Software Guard extensions (SGX) provides "enclaves"**
  - Newer, more "nimble" TXT
- **Intel system management mode (SMM) is a stealthy execution environment ("ring -2") for chipset manufacturer code to live**
  - Fully hidden in HW from OS/hypervisor
  - Can host a 2nd VMM that virtualizes chipset code (DMM)

Interesting Research

Private key side-channel leakage

- **VT-x may provide strong isolation, but side-channels still exist**
  - Timing
  - Shared processor cache
  - IO

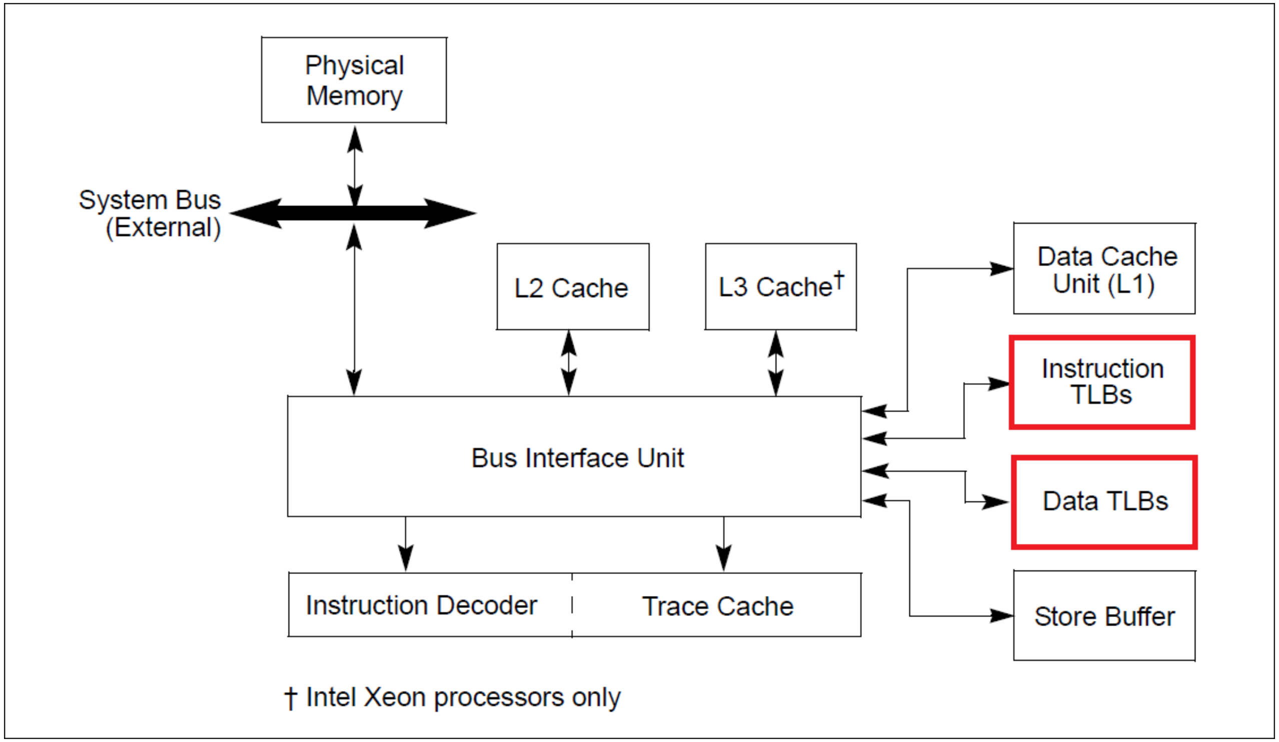
- **Extracted a private key being used in a VM from another co-resident VM on the Xen hypervisor**
  - Used cache timing (similar to AES attack) to figure out what memory other VM was accessing
  - Able to recover ElGamal private key in lab setting

- **Reminder that VT-x is not designed for total isolation**

Technical Background

TLB

- **TLB is physically two separate entities, one for code, one for data**



Interesting Research

Measurement of running executables

- Built upon GRSecurity’s PAGEEXEC & Shadow Walker rootkit to split TLB to provide periodic measurements of dynamic code applications
- Transparently segregates code and data fetches to different regions of memory
- Can detect code-injection attack almost instantly
- DARPA Cyber Fast Track effort

Interesting Research

TRESOR

- **Used AES-NI in a Linux kernel patch to support on-CPU AES**
  - Protects against cold-RAM attacks

- **Key stored in protected debug registers**
  - GDB, etc. will resort to software breakpoints (INT3)

- **Loaded at early boot (single-user mode)**
  - Less programs running to snoop

- **TRESOR-Hunt shows weaknesses to DMA attacks**

Concluding Remarks

- **Hopefully this provided enough of an overview of Intel VT-x for you to feel confident to play with it**

- **I have a simple hypervisor which I built on for MoRE for anyone who is interested**
  - Windows 7 x86 kernel driver
  - Loads thin hypervisor into VMM slot
  - https://github.com/ainfosec/more

- **Don’t hesitate to contact me with questions or to bounce ideas around**

Questions?

- Bedankt!