MORE SHADOW WALKER: THE PROGRESSION OF TLB-SPLITTING ON X86

Jacob Torrey: Cyber-security Philosopher
@JacobTorrey

About Me

- Senior research engineer at Assured Information Security (ainfosec.com)
- Leads low-level computer architectures group
  - SMM
  - VMM
  - BIOS
  - Kernel
- Lover of the great outdoors

Pre-talk Notes

- Opinions are my own, I don't speak for:
  - AIS
  - DARPA
  - GRSecurity
  - Jamie Butler

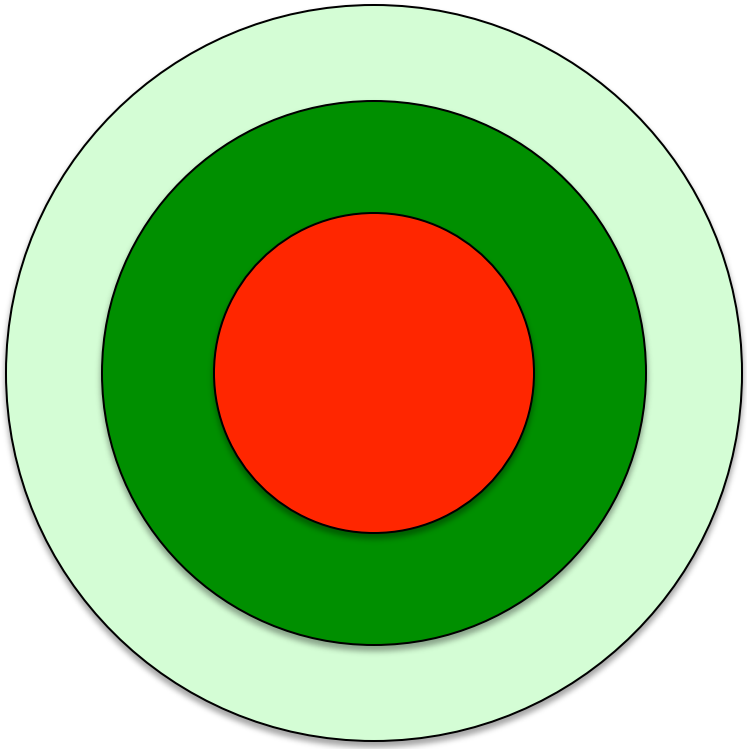
- **PLEASE** interrupt (politely) with questions if something is unclear

Outline

- Background
- History
  - PaX/GRSecurity - Defensive
  - Shadow Walker - Offensive
  - MoRE - Defensive
  - MoRE Shadow Walker - Offensive
- Conclusions
- Wrap up

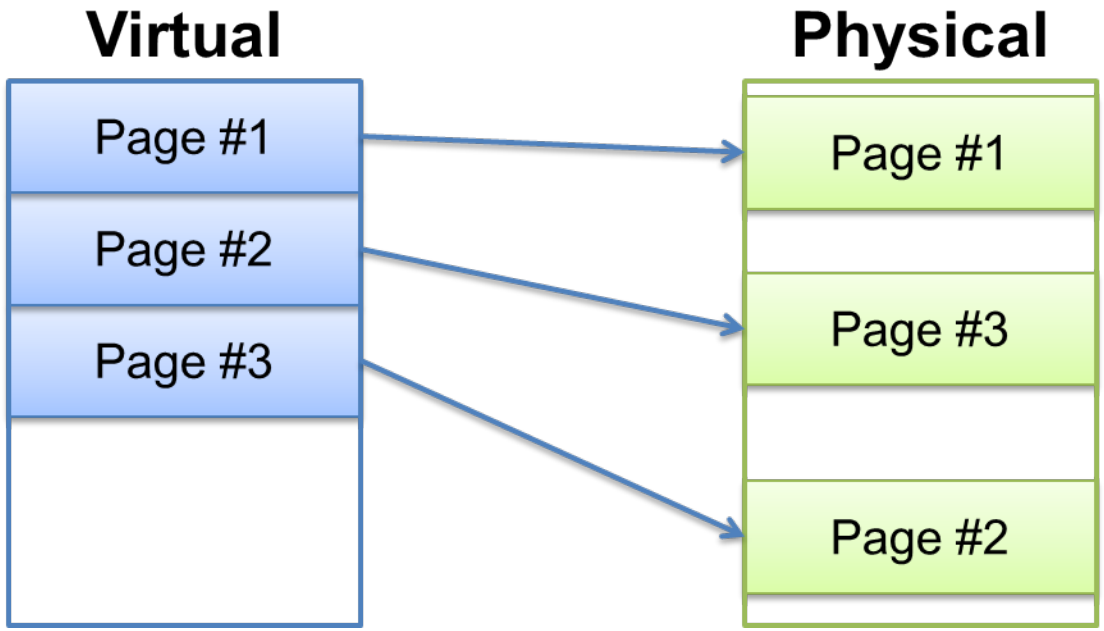
x86 Protection Rings

- "Rings" of protection:
- Ring 3 – User apps
- Ring 1 & 2 – nada
- Ring 0 – OS kernel
- Ring -1 – Hypervisor



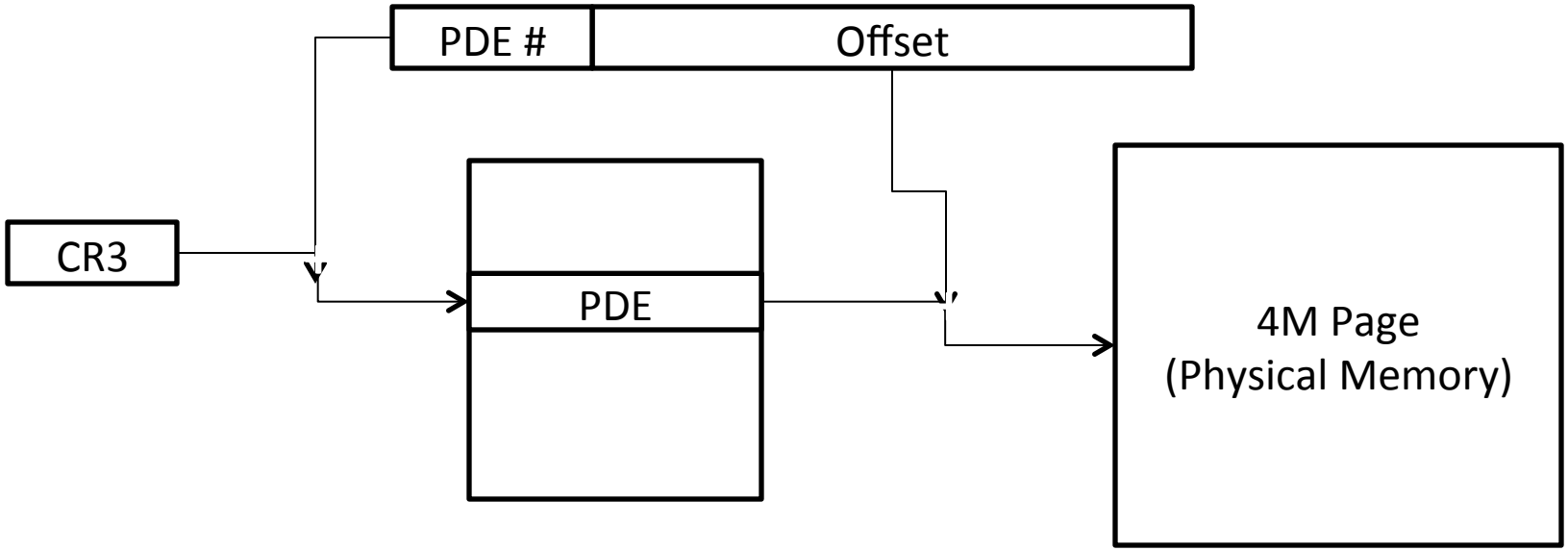
Background
Virtual Memory

- Intel x86 provides OS method to abstract view of memory: virtual memory / paging



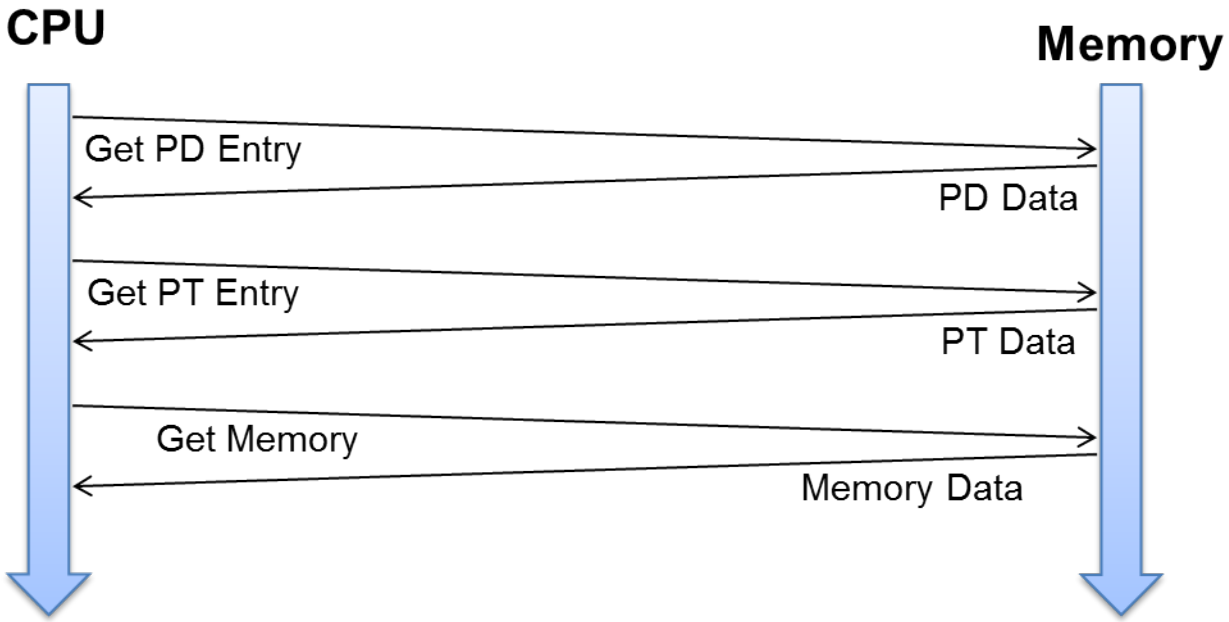
Background
Address Translation

- The virtual address consists of array indices and an offset:



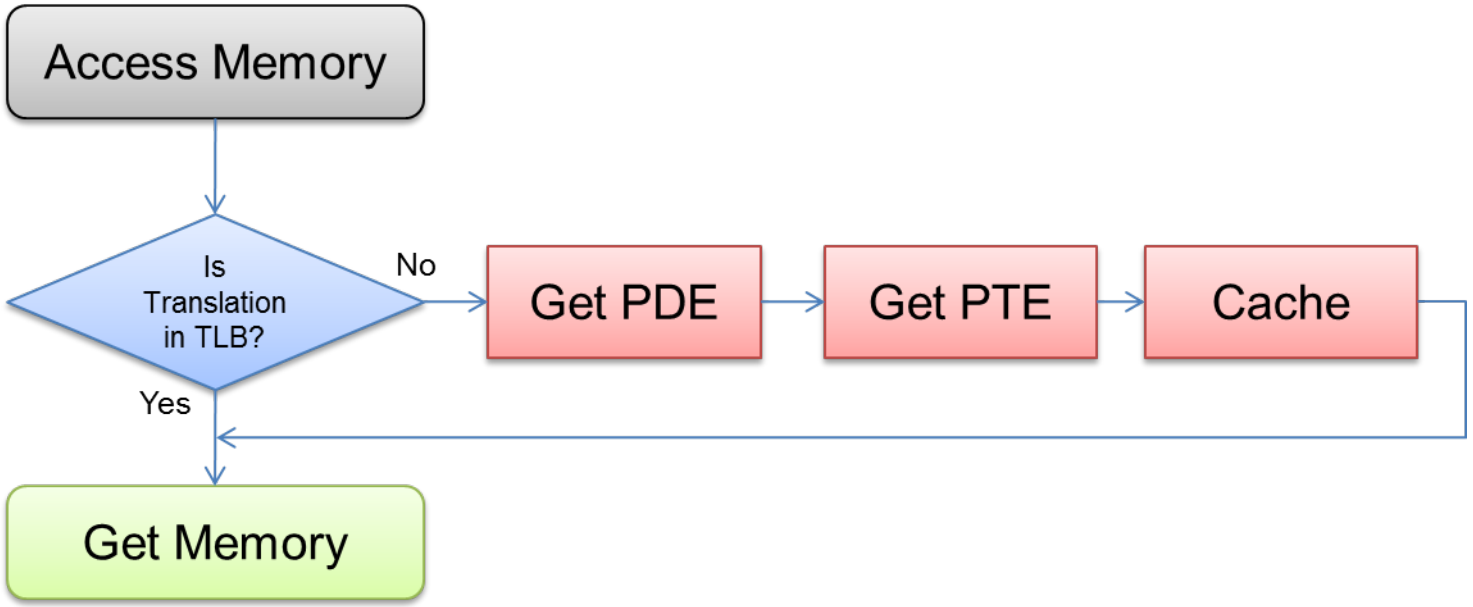
Background
Page Translation

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



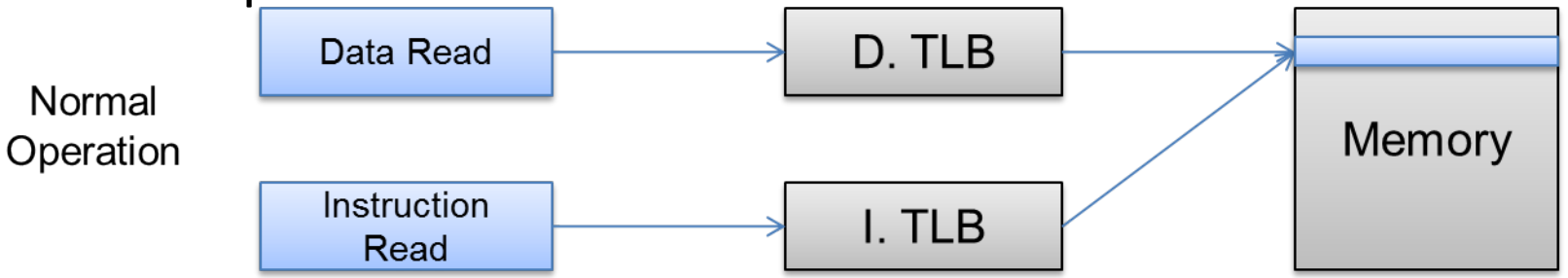
Background
Translation Lookaside Buffer

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

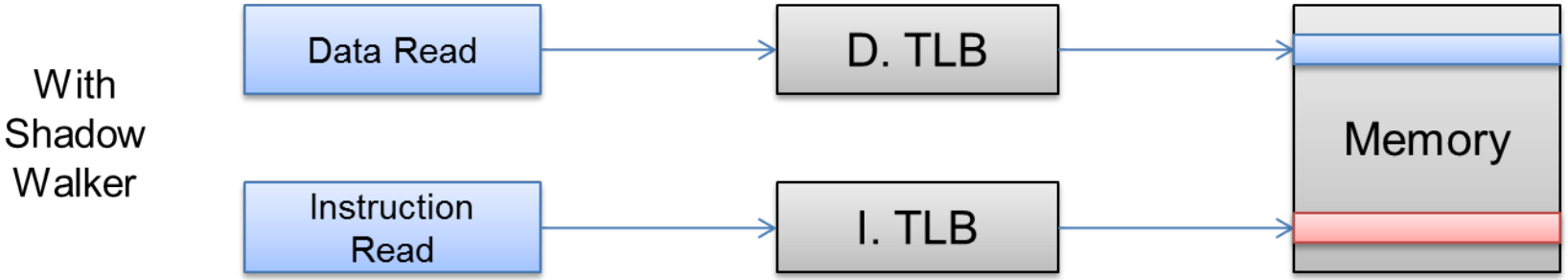


Background
TLB Splitting

- The CPU's TLB is used to cache memory page translations to increase performance.

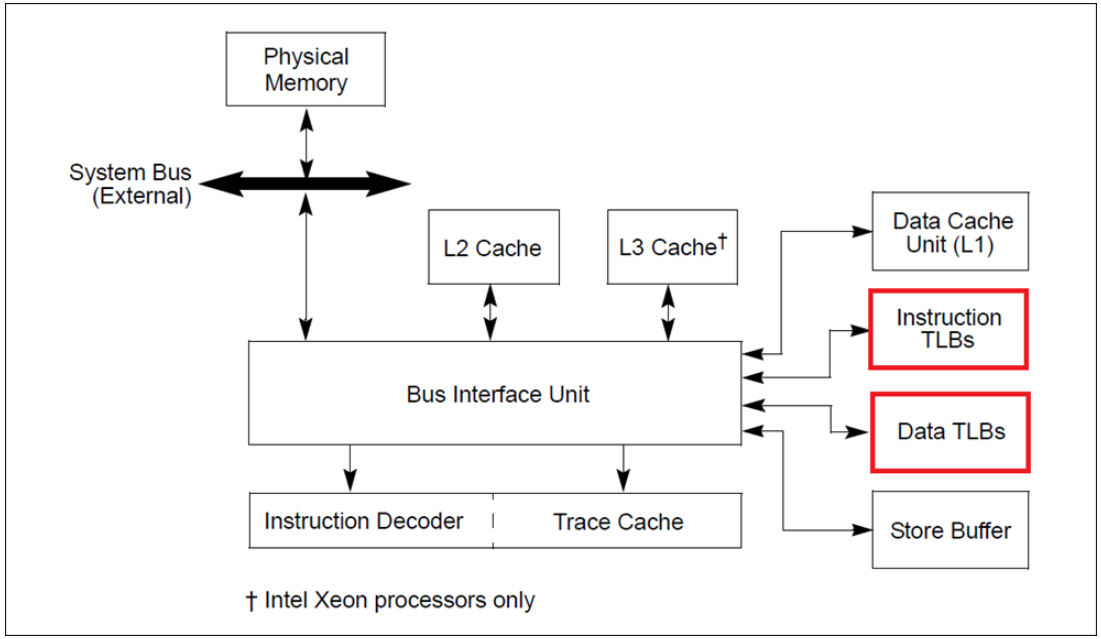


- De-synchronizing a CPU's Translation Look-aside Buffer (TLB) (e.g. Shadow Walker or PaX).



Background
Intel TLB

- In pre-Nehelam CPUs, the D-TLB and I-TLB were completely separate:



Splitting the TLB

- On a pre-Nehelam CPU, the TLB is split by:
  - Trapping on a fault (page or protection)
  - Determining if data or instruction (check EIP and CR2 or exit condition)
  - "Priming" the proper TLB by either setting data access bit(s) and reading then clearing bit(s) (but not TLB) or allowing 1 instruction to go (trap flag)
  - Repeat as needed

Questions Yet?

History

PaX (2000) - Defensive

- PaX Team - PAGEEXEC
- Set of Linux kernel patches to emulate no execute (NX) bit on earlier x86 CPUs without NX support
- Uses user/supervisor bit as NX bit, and splits the TLB for performance gains (data entries are cached)
- Works on pre-Nehelam CPUs

History

Shadow Walker (2005) - Offensive

- Jamie Butler and Sherri Sparks
- Windows memory-hiding rootkit
- Hooks page fault handler to split the TLB and hide kernel rootkit pages
  - FU rootkit
  - Prevents detection by anti-virus, etc.
- Works on pre-Nehelam CPUs

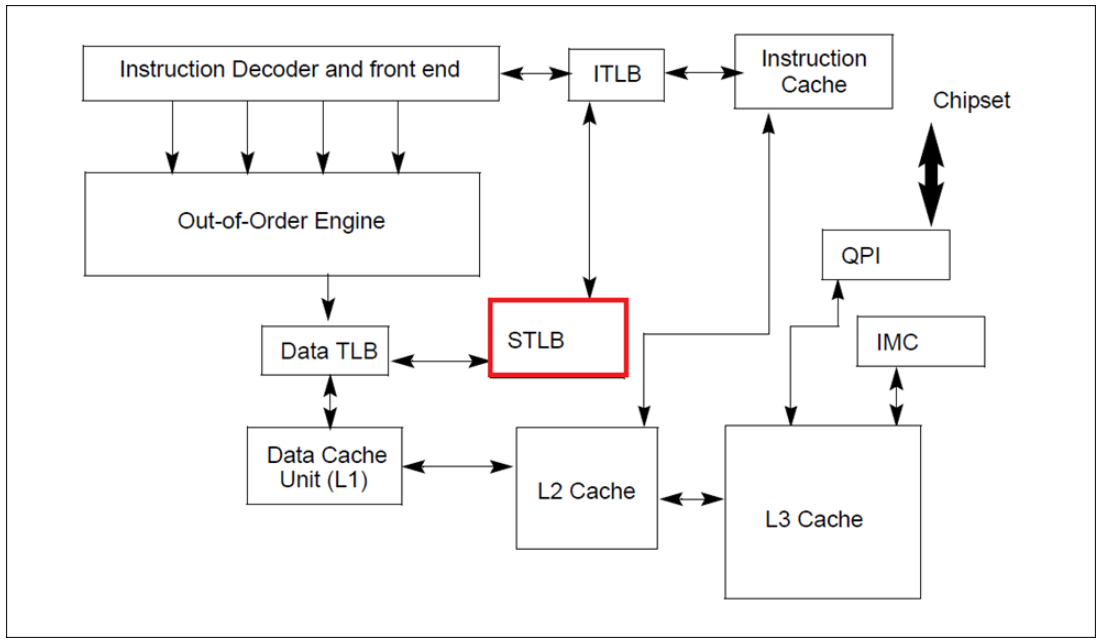
End of an Era

Intel breaks TLB-splitting

- Intel releases the Nehelam architecture (1st generation Core i-series)
- Addition of a level 2 cache for TLB, a shared TLB, or S-TLB
- Previous TLB splitting tools will not work due to this major architecture change
  - Hangs in endless loop as S-TLB merges entries
  - Not enough permission granularity

Background
Intel S-TLB

- After Nehelam, Intel introduced the shared TLB (S-TLB):



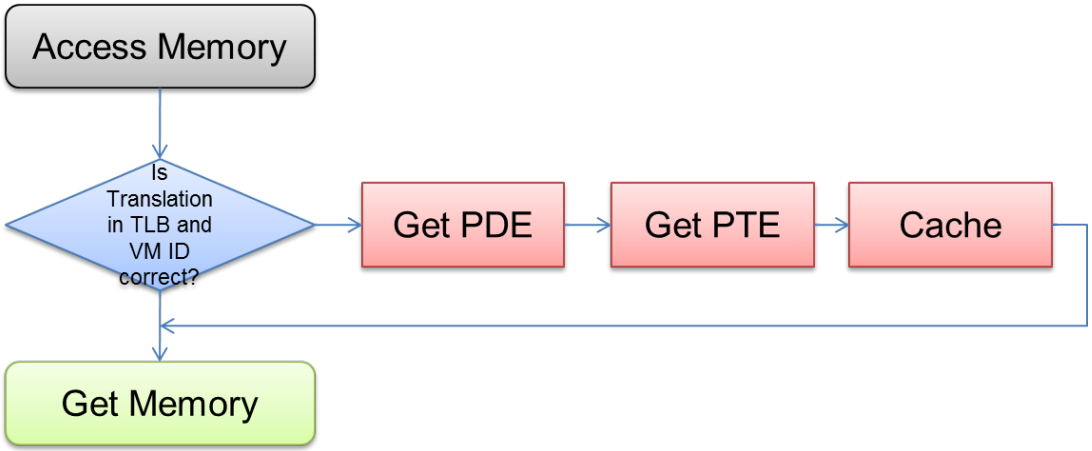
Background
Intel EPT & VPID

- 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
  - **Supports execute-only permission bit**

- 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

Background
Intel VPID

- Along with EPT, the TLB was extended to prevent the performance hit of virtualization:

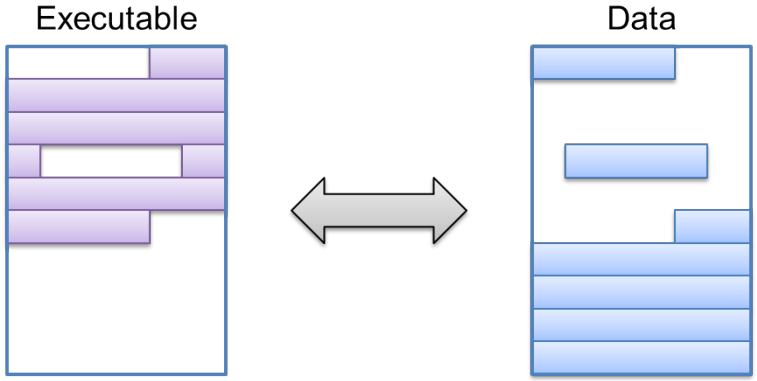


Splitting the TLB

- On a Nehelam+ CPU, the TLB is split by:
  - Setting up EPT tables; trapping on EPT faults
  - Determining type of fault
  - Setting R/W or E bits in EPT
  - Setting trap to VMM bit
  - Run 1 instruction to prime proper TLB VPID
  - Reset EPT to E/R/W = 0
  - Resume guest

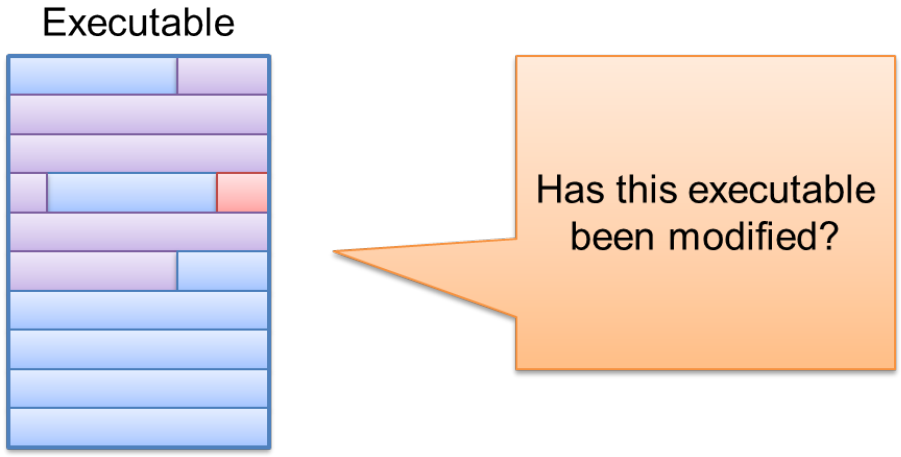
MoRE
Hypothesis

- We believe that the same TLB de-synchronization used by Shadow Walker can be used to automatically separate data references from already existing applications in real-time for real-time trust measurements



MoRE
Goal

- The DARPA CFT MoRE program sought to identify if TLB splitting could be used to detect application subjugation even if an executable's data and code are mixed



MoRE
Challenges

- Intel's Nehalem S-TLB
- Windows 7 Memory Manager
- TLB is cleared on VMEXIT
- Windows Copy-on-write

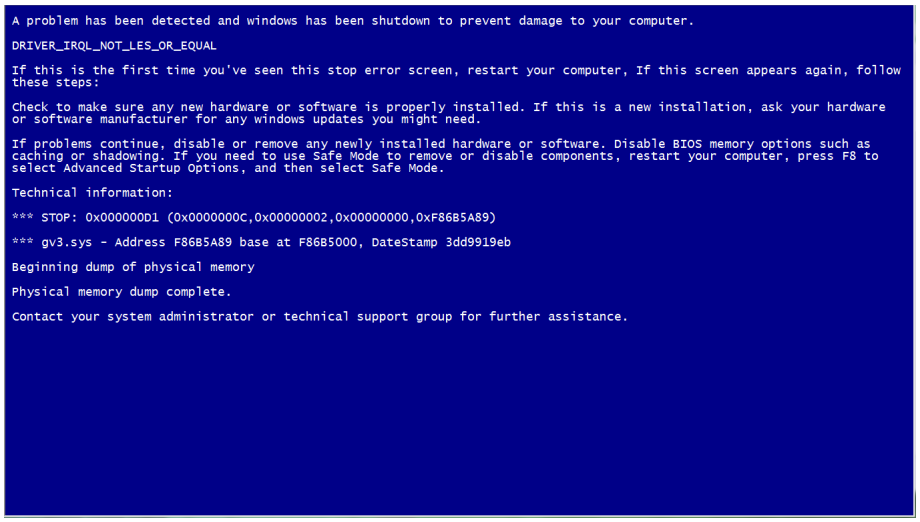
Intel S-TLB
Challenges

- Luckily Intel created a way around the S-TLB: Extended Page Tables (EPT)
- EPT is hardware-assisted VM memory management
  - Allows a hypervisor to create page tables mapping “guest physical addresses” to “machine physical addresses”
- EPT supports more granular permissions than standard paging (E/R/W)
  - The S-TLB will not merge entries with conflicting permissions for security

Windows 7 Memory Manager

Challenges

- Shadow Walker also focused on hiding kernel pages, many of which are rarely (if ever paged out)
- When you muck with the paging structures for user-space applications, Windows does not approve



Windows 7 Memory Manager

Challenges

- What to do…
- EPT to the rescue!
  - The hypervisor can modify the EPT page tables without the OS noticing and preempt any faults (fixing things before the OS wizens up)
- Thanks again Intel for providing a solution to our problems!

TLB Clear on VM Exit

Challenges

- Now that MoRE has moved from a kernel page-fault hook (ala Shadow Walker) to a hypervisor to bypass those problems, a new problem arises
- The TLB is cleared when VM EXITing/VM RESUMEing to prevent guests from reading hypervisor memory
- Hard to prime a TLB if it gets cleared!

TLB Clear on VM Exit

Challenges

- Intel to the rescue! VPID (tagged TLB entries)
- Thanks Intel!

Windows COW
Challenges

- Windows loads 1 copy of code into memory for each application, even if multiple instances are running
  - Performs copy-on-write to only duplicate code for changed pages
- Our test application interleaved code and data
  - Always triggering COW
- We would set up the split and the initial write would trigger Windows to move our application out from under us!
  - TLB splitting was now on unused pages, where did our app go?
- Simple, yet ugly solution:
  - VM EXIT on CR3 change (context switch)
  - Walk kernel page tables to see if COW occurred and update list of application pages

MoRE
Challenges

- Intel's Nehalem S-TLB
  - The MoRE VMX hypervisor uses extended page tables (EPT) to create different permission sets to prevent TLB merging
- Windows 7 Memory Manager
  - VMX hypervisor with EPT bypasses Windows memory manager
- TLB is cleared on VMEXIT
  - MoRE implements VPID tagged TLB entries for TLB persistence
- Windows Copy-on-write
  - MoRE monitors page table modifications for target applications upon CR3 change

MoRE
Design

- Built a custom VMX hypervisor with EPT and VPID support that could monitor process creation
- Would VMEXIT on:
  - CR3 change – Monitor process change for Windows COW activity
  - EPT violations – Essentially the same as a page fault to the hypervisor
  - Trap Flag – Let us prime the TLB entry with the correct VPID
- Like Shadow Walker, we either:
  - If data access – Mark EPT as R/W for 1 instruction
  - If code access – Mark EPT as E for 1 instruction
  - If both – Special case, copy 1 instruction to data copy, run as R/W/E then reset to no permissions
    - Including TLB flush

MoRE
Results

- Even with the prototype nature of MoRE, performance hits were <2%
- Could perform periodic measurements of an application and the MoRE system (designed to be measurable) very rapidly – re-verifying trust every <1/10th of a second!
- Required no modification of application, no recompilation or source

MoRE Video

MoRE Shadow Walker

Swinging back to the offensive

- TLB-splitting is just a technique – clear that is can be used for both offense and defensive
- MoRE Shadow Walker is a modification to MoRE that allows memory hiding even from ring 0 code
  - Patch Guard?
- Can split on arbitrary pages on Nehelam and newer CPUs

MoRE Shadow Walker

Why?

- Why bother?
  - Can just use VMM to protect root kit
- Higher granularity of control, can run and hook kernel functions
- VMM shim is very small to minimize performance hit
- To show that technology has no default good/ evil, depends on intent if you are a white-hat or “merchant of death”

Conclusion

- The immense complexity of the Intel x86 ISA enables huge architectural modifications to be effected *through software*
  - Ex: Turing-complete MMU
- Even as architecture evolves, so too does the techniques to misuse it
  - Ex: NX bit

The code

- The code for a simple TLB splitting VMM (for Windows 7) can be found on AIS's Github repository:
  - http://github.com/ainfosec/MoRE
- Released this week!

Shout outs

- @grsecurity & PaX team for helping make Linux more secure
- @jamierbutler for helping provide guidance on the CFP submission
- @dotMudge and @DARPA for taking MoRE from proposal to implementation
- @ainfosec for putting up with me and my long periods of dropping off the face of the earth

Questions?

- Thanks for listening!
  - Let the heckling begin
  - Rotten tomatoes?
- I'm here all week, tweet @JacobTorrey if you want to meet up later
  - To the bar I go now!