Pitfalls of virtual machine introspection on modern hardware

Tamas K. Lengyel
@tklengyel
tamas@tklengyel.com

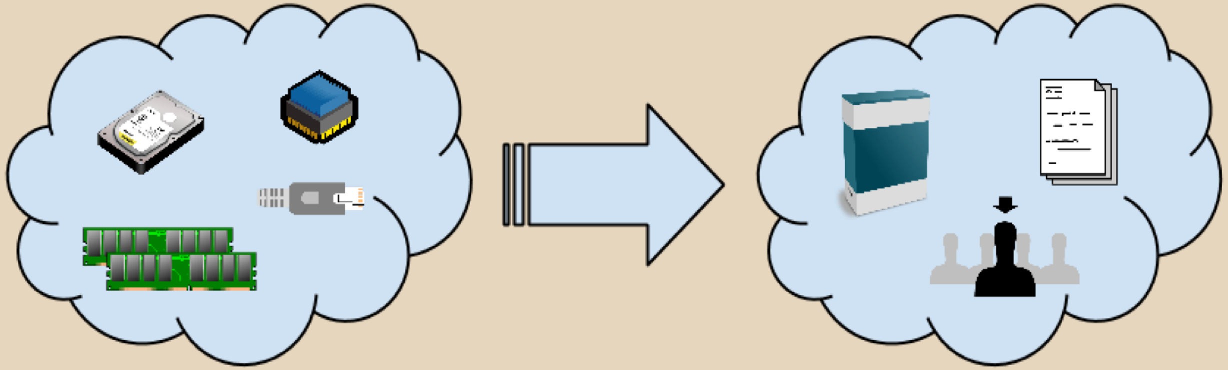
Agenda

1. VMI intro
2. Software attacks
   - Direct Kernel Structure Manipulation
   - Direct Kernel Object Manipulation
3. Hardware attacks
   - Translation Lookaside Buffer poisoning
   - Extended Page Tables limitations
   - System Management mode
4. Conclusion

Virtual Machine Introspection (VMI)

Interpret virtual hardware state

- Network, Disk, vCPU & Memory
- The semantic gap problem:
  Reconstruct high-level state information from low-level data-sources.



Bridging the semantic gap

- The guest OS is in charge of managing the virtual hardware
  - How to get the info from it?
- Install in-guest agent to query using standard interfaces
  - If OS is compromised in-guest agent can be disabled / tampered with
  - Just as vulnerable as your AntiVirus

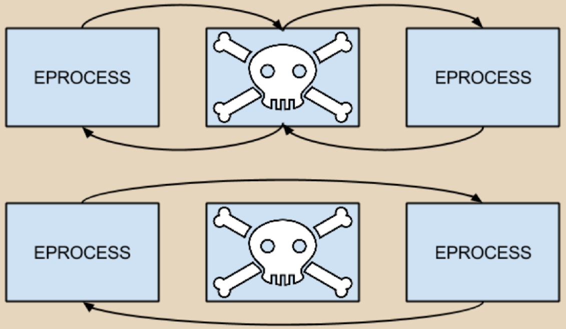
Bridging the semantic gap

- Replicate guest OS functions externally
  - In-guest code-hooks are avoided
  - Requires expert knowledge on OS and hardware behavior
  - Requires debug data to understand in-memory data-structures

- This is where the problems begin
  - The weak and the strong semantic gap

Direct Kernel Object Manipulation

2004: DKOM - Mangle in-memory data-structures to hide elements



LibVMI example:

```
size_t vmi_read_va(
      vmi_instance_t vmi,
      addr_t vaddr,
      vmi_pid_t pid,            // This is interpreted by
                                // walking the list
      void *buf,
      size_t count);
```

Direct Kernel Structure Manipulation

DKSM: Subverting Virtual Machine Introspection for Fun and Profit (2010)

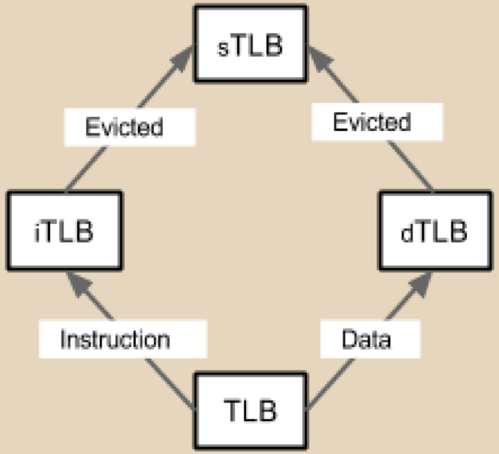
- Patch in-guest system to interpret structures differently

External interpretation	Internal interpretation

typedef struct _FOO
{
    void* my_pointer;
    unsigned long my_value;
} FOO;
	typedef struct _NEWFOO
	{
	    unsigned long my_value;
	    void* my_pointer;
	} NEWFOO;

Translation lookaside buffer (TLB) poisoning

- Virtual to physical address translation is expensive
- Hardware managed transparent cache of the results
- Separate cache for read/write and instruction fetch (Harvard-style architecture)!
- Opportunity to whack it out of sync!
- Shadow Walker / FU rootkit



TLB poisoning

Original algorithm:

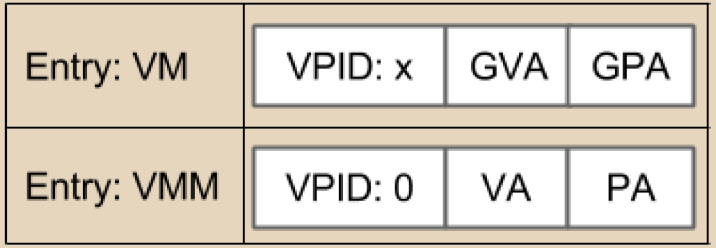
Input: Splitting Page Address (*addr*)
Pagetable Entry for addr (*pte*)

```
invalidate_instr_tlb (pte);          // flush TLB
pte = the_shadow_code_page (addr); // replace PTE in memory
mark_global (pte);                   // disable auto-flush
reload_instr_tlb (pte);              // load it into TLB
pte = the_orig_code_page (addr);   // put original entry back
```

TLB poisoning and virtualization

Automatically flush of the TLB on every VMEXIT/VMENTRY

- TLB (poisoning) is impossible
- Performance hit



Introduction of TLB tagging (VPID) in Intel Nehalem (2008)

- 16-bit field specified in the VMCS for each vCPU
- Performance boost!
- VM TLB entries invisible to the VMM
- The problem is not (just) the split TLB

TLB poisoning with Windows / Linux

TLB poisoning uses *global pages*

- CR4.PGE (bit 7)
- Makes PTE’s marked as global survive context-switches (MOV-TO-CR3)
- Great performance boost for kernel pages!

Windows 7

- Regularly flushes global pages by disabled & re-enabling CR4.PGE

Linux

- Doesn’t touch CR4.PGE after boot

The tagged TLB in Xen

- The TLB tag is assigned to the vCPU from a global counter

  ```
  asid->asid = data->next_asid++;
  ```
- No flushes, just assign a new tag when needed
- When 16-bit field is exhausted, flush and start from 1
- A new tag is assigned on every MOV-TO-CR3
  - The use of global pages disabled in the guest!
  - The TLB needs to be primed on each context-switch

The tagged TLB in KVM

- Tag is assigned when vCPU structure is created
  - Doesn’t matter if the vCPU is activated or not
  - Ran out of assignable tags?
    - Disable tagging and revert to old VMENTRY/VMEXIT TLB flushing
- Priming the TLB in Linux guests on KVM is a problem for VMI
- However, the split TLB still has issues

The sTLB!

Intel Nehalem introduced second-level cache: sTLB

The problem:

- Split-TLB relies on a custom page-fault handler being called to re-split the TLB when it’s evicted
- With sTLB, the entry is brought back into the 1st level cache
  - ..both into the iTLB & the dTLB
  - Split-TLB becomes unsplit!
- Split-TLB poisoning is unreliable in VMs!

MoRE Shadow Walker

2014: The evolution of TLB splitting on x86

- Guests can’t disable the sTLB by themselves
- However, sTLB doesn’t merge entries with conflicting PTE permissions
  - 1st level PTEs can have R, R+W or R+E permissions
  - 2nd level (EPT) PTEs can have R, R+W or E permissions!
- Reliable TLB splitting requires VMM support!

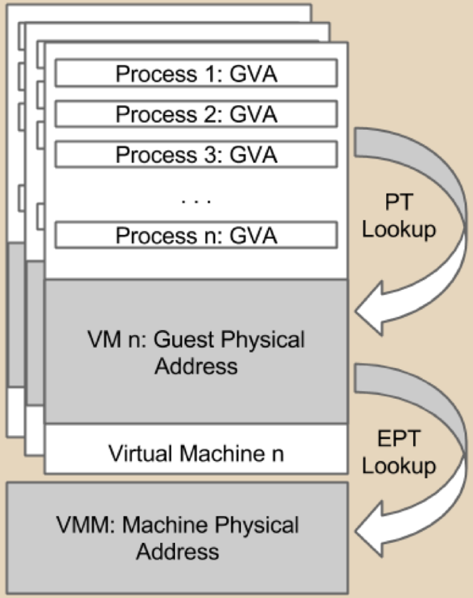
Extended Page Tables (EPT)

Speed up guest virtual to machine physical address translation

Two sets of tables

- 1st layer managed by guest OS
- 2nd layer managed by the VMM

Permissions can be different in the two layers!



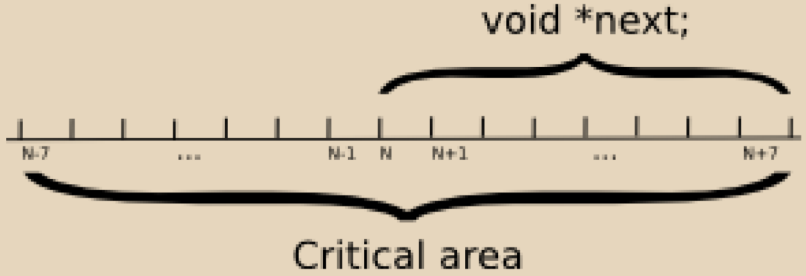
Extended Page Tables (EPT)

Can be used to trace the execution and memory accesses made by the guest

- Transparently
- 4k Page-level granularity at best
- Need to filter unrelated events!

Notable commercial examples:

EPT limitations



- Only the start address and type of the violation is recorded
- We don't know how much memory is involved
- The operating system by default starts R/W operation at the start of a variable
- But it is not enforced
- Violations in the vicinity of a watched area need to be treated as potential hits

EPT limitations

Read/Write violation ambiguities

"An EPT violation that occurs during as a result of execution of a read-modify-write operation sets bit 1 (data write). Whether it also sets bit 0 (data read) is implementation-specific and, for a given implementation, may differ for different kinds of read-modify-write operations."

Intel SDM

“Counter question: Why can't the hardware report true characteristics right away?”

Jan Beulich – SuSE

“when spec says so, there is a reason but I can't tell here. :-)”

Kevin Tian – Intel

EPT limitations

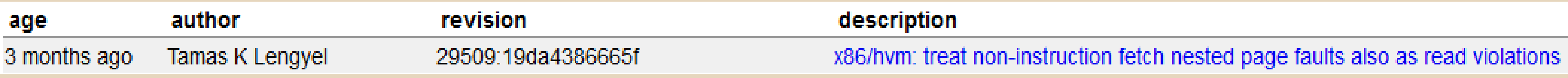
Read/Write violation ambiguities

"An EPT violation that occurs during as a result of execution of a read-modify-write operation sets bit 1 (data write). Whether it also sets bit 0 (data read) is implementation-specific and, for a given implementation, may differ for different kinds of read-modify-write operations."

Intel SDM

It is possible to siphon data using r-m-w operations from a page that doesn’t allow reading!

Fixed in Xen 4.5



age	author	revision	description
3 months ago	Tamas K Lengyel	29509:19da4386665f	x86/hvm: treat non-instruction fetch nested page faults also as read violations

EPT limitations

- Single EPT per guest
  - Not a hardware limitation
  - Could have separate EPT for each vCPU
- Tracing with multi-vCPUs
  - EPT permissions need to be relaxed while one vCPU is advanced
  - Race condition
  - All vCPUs need to be paused while one vCPU is singlestepped

System Management Mode (SMM)

SMM intended for low-level services, such as:

- thermal (fan) control
- USB emulation
- hardware errata workarounds

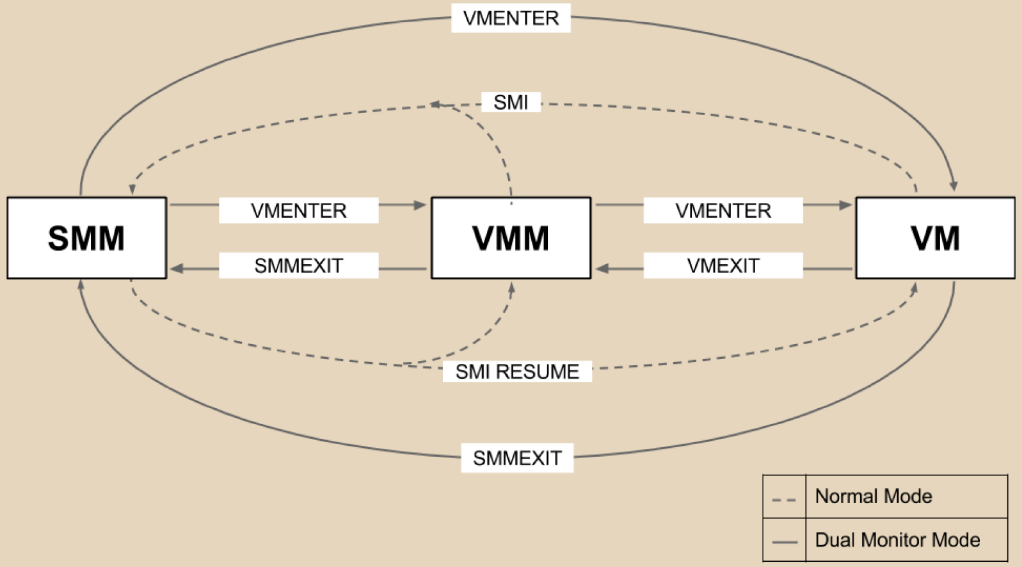
Can be used for:

- VMI
- Anti-VMI!

Hard to take control of it on (most) Intel devices as it is loaded by the BIOS

SMM

- Normal mode SMM is triggered by interrupts (SMI)
- Can be configured to happen periodically
- Always returns to the same execution mode afterwards



SMM

The problem for SMM based VMI systems:

"A limitation of any SMM-based solution [...] is that a malicious hypervisor could block SMI interrupts on every CPU in the APIC, effectively starving the introspection tool. For VMI, trusting the hypervisor is not a problem, but the hardware isolation from the hypervisor is incomplete."

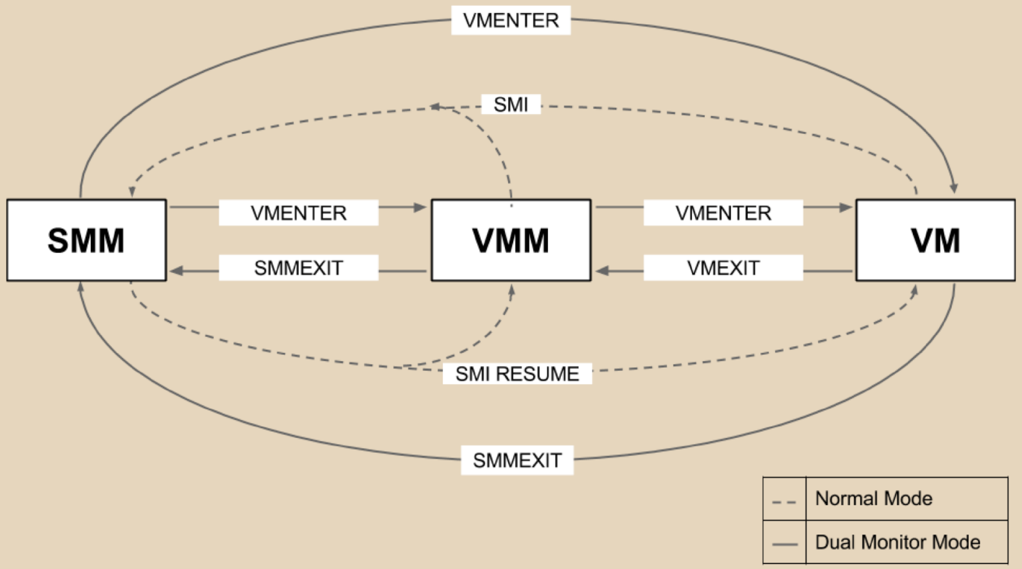
Jain et al. "SoK: Introspections on Trust and the Semantic Gap" 2014, IEEE S&P

Intel Dual-monitor mode SMM

- Available on all CPUs with VT-x (?)

- SMM can become an independent hypervisor

- VMCALL in VMX-root!



Intel Dual-mode SMM

The VMCALL instruction can be used to instrument the VMM

- Same way INT3 can be used to instrument a VM
- Starvation is impossible via the APIC

The SMM can enter any execution mode

- Full control over the execution flow
- Hidden VMs

The SMM can temporarily disable SMIs for a VM!

- Forced execution

Conclusion

VMI is powerful but has issues

- The strong semantic gap

Hardware support is better

- Tagged TLB is a problem
- Split-TLB requires VMM support
- EPT corner-cases need to be taken into consideration

Dual-mode SMM is un(der)-explored