Abusando da Virtualização

Gabriel Negreira Barbosa

gabriel.negreira.barbosa [arroba] intel.com

Importante

- Eu não falo pelo meu empregador
  - Todas as ideias e informações presentes nessa apresentação são de minha inteira responsabilidade

- Não espere o final da palestra para fazer perguntas!

Introdução

- Virtualização cada vez mais utilizada:
  - Sistemas operacionais modernos
  - Malware analysis
  - Cloud
  - Etc

- CPUs possuem recursos poderosos para um melhor uso da virtualização
  - Como resultado, malware executando nas camadas mais baixas dessa tecnologia também podem utilizar tais recursos

Objetivos

- Essa apresentação NÃO vai discutir como quebrar sistemas de virtualização

- Objetivos principais:
  - Mostrar que, além de ser divertido, fazer experimentos com virtualização pode também ajudar bastante os estudos de conceitos computacionais
  - Discutir ideias sobre o que um atacante que já possua acesso às camadas mais baixas de virtualização pode fazer
    - O foco não é prover uma lista completa de técnicas, mas uma visão geral sobre algumas ideias
    - Todas as ideias podem também ser aplicadas para proteger sistemas

Pré-requisitos

- Intel VMX
- Linux 64-bit
- Conhecimentos básicos de C, C++ e assembly Intel

- Bareflank (https://github.com/Bareflank) foi utilizado como base para os demos

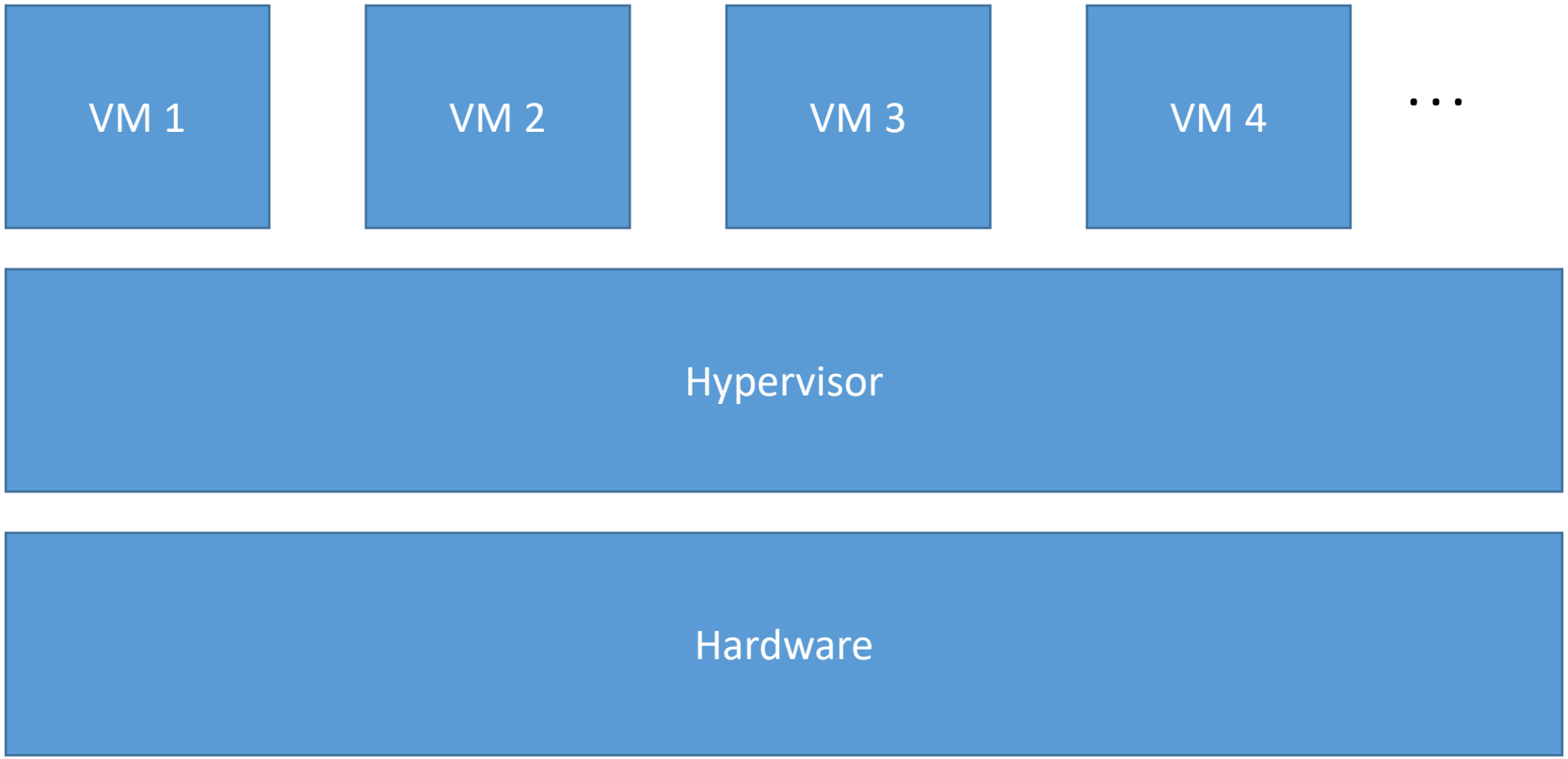
Agenda

- Conceitos de virtualização
- Abusando da virtualização
- Conclusões

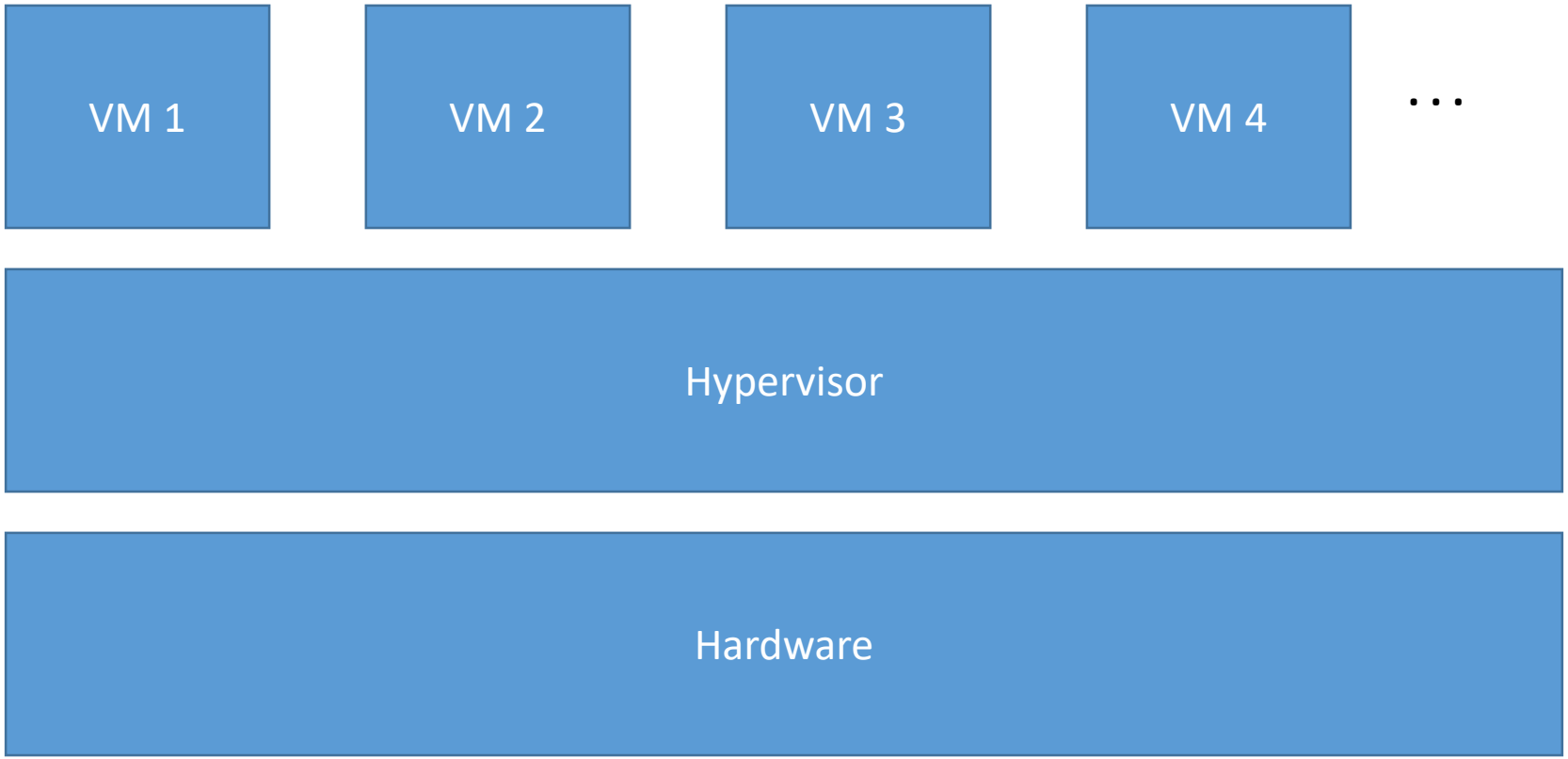
Agenda

- Conceitos de virtualização
- Abusando da virtualização
- Conclusões

Arquitetura em Alto Nível



Arquitetura em Alto Nível

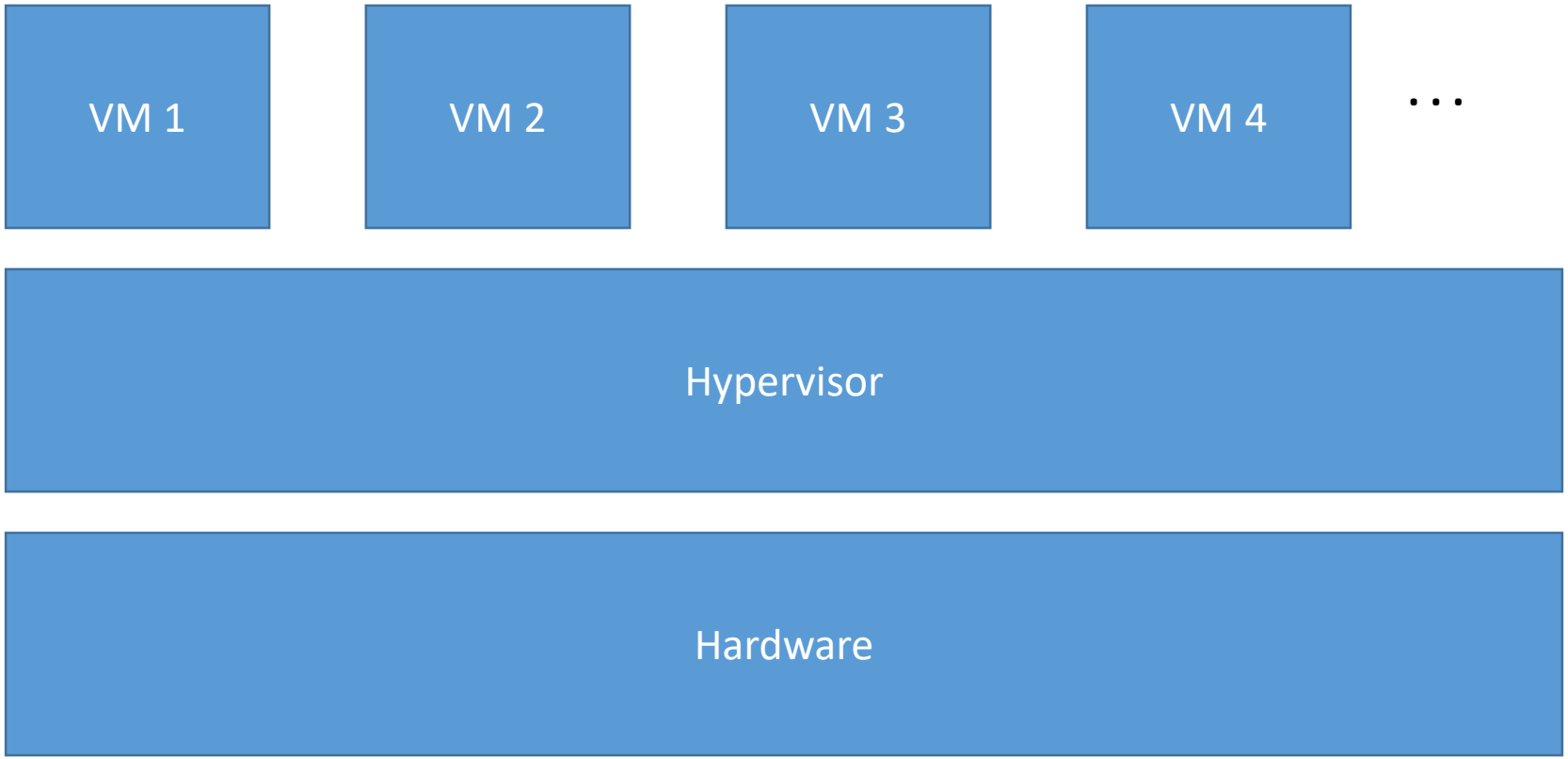


Onde está Windows?

VMX (Virtual-Machine Extensions)

- VMX Root
  - Comportamento muito parecido como não-VMX
    - Exemplo de diferença: instruções VMX
  - Em geral, VMM (Virtual Machine Monitor) / **hypervisor** roda como VMX root
- VMX Non-Root
  - Ambiente com restrições e modificações para facilitar a virtualização
    - Exemplos ao decorrer da apresentação
  - Em geral, **máquinas virtuais** rodam como VMX non-root

Arquitetura em Alto Nível



Como o hypervisor controla as VMs?

VM Entry and VM Exit

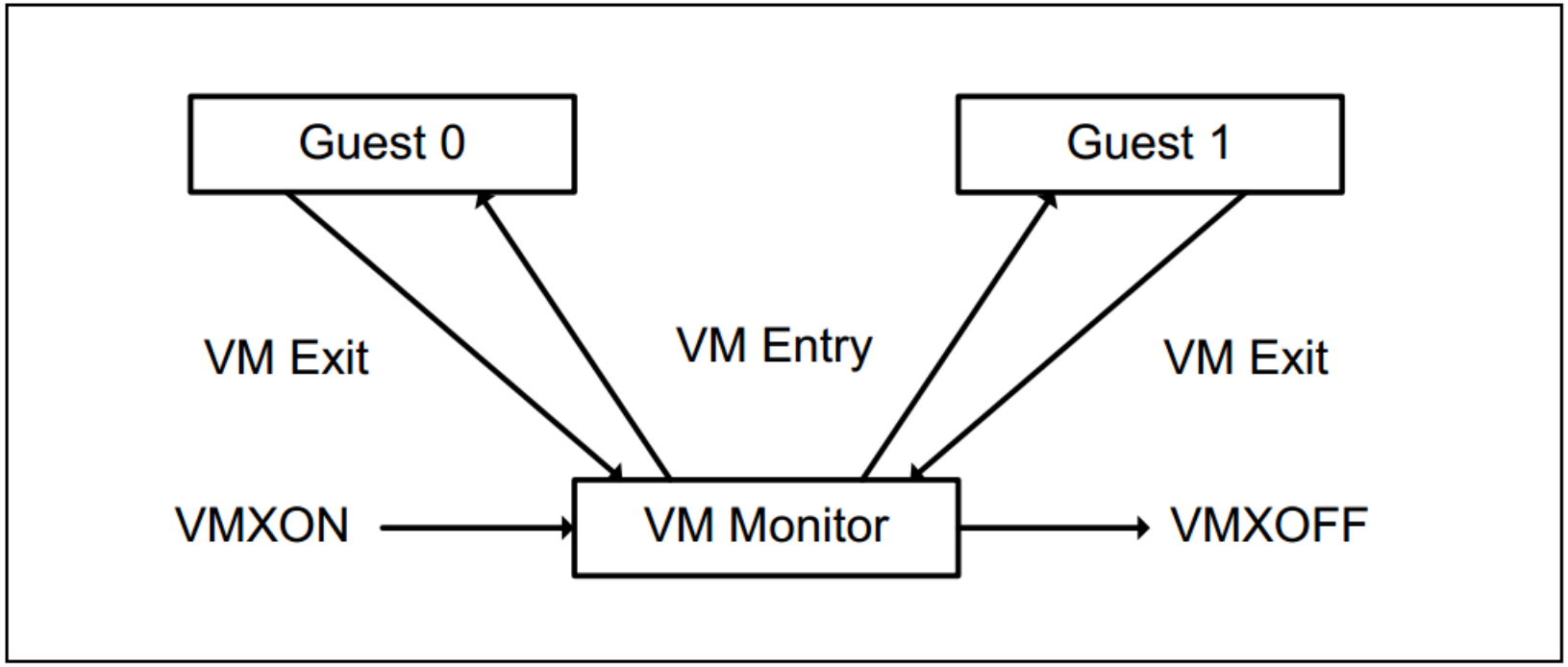


Figure 23-1. Interaction of a Virtual-Machine Monitor and Guests

Intel VMCS (Virtual Machine Control Structure) 1/2

- Estrutura utilizada pelo processador durante operação VMX
- Utilizada para gerenciar o comportamento e transições (VM entries and VM exits) do VMX non-root.
- Manipulada pelas instruções VMCLEAR, VMPTRLD, VMREAD, and VMWRITE

Intel VMCS (Virtual Machine Control Structure) 2/2

Table 24-1. Format of the VMCS Region

Byte Offset	Contents
0	Bits 30:0: VMCS revision identifier Bit 31: shadow-VMCS indicator (see Section 24.10)
4	VMX-abort indicator
8	VMCS data (implementation-specific format)

24.3 ORGANIZATION OF VMCS DATA

The VMCS data are organized into six logical groups:

- **Guest-state area.** Processor state is saved into the guest-state area on VM exits and loaded from there on VM entries.
- **Host-state area.** Processor state is loaded from the host-state area on VM exits.
- **VM-execution control fields.** These fields control processor behavior in VMX non-root operation. They determine in part the causes of VM exits.
- **VM-exit control fields.** These fields control VM exits.
- **VM-entry control fields.** These fields control VM entries.
- **VM-exit information fields.** These fields receive information on VM exits and describe the cause and the nature of VM exits. On some processors, these fields are read-only.[3]

The VM-execution control fields, the VM-exit control fields, and the VM-entry control fields are sometimes referred to collectively as VMX controls.

Virtualização da CPU – Exemplo 1/2

25.1.2 Instructions That Cause VM Exits Unconditionally

The following instructions cause VM exits when they are executed in VMX non-root operation: CPUID, GETSEC,[1] INVD, and XSETBV. This is also true of instructions introduced with VMX, which include: INVEPT, INVVPID, VMCALL,[2] VMCLEAR, VMLAUNCH, VMPTRLD, VMPTRST, VMRESUME, VMXOFF, and VMXON.

25.1.3 Instructions That Cause VM Exits Conditionally

Certain instructions cause VM exits in VMX non-root operation depending on the setting of the VM-execution controls. The following instructions can cause "fault-like" VM exits based on the conditions described:[3]

...

- **RDMSR.** The RDMSR instruction causes a VM exit if any of the following are true:
  - The "use MSR bitmaps" VM-execution control is 0.
  - The value of ECX is not in the ranges 00000000H – 00001FFFH and C0000000H – C0001FFFH.
  - The value of ECX is in the range 00000000H – 00001FFFH and bit *n* in read bitmap for low MSRs is 1, where *n* is the value of ECX.
  - The value of ECX is in the range C0000000H – C0001FFFH and bit *n* in read bitmap for high MSRs is 1, where *n* is the value of ECX & 00001FFFH.

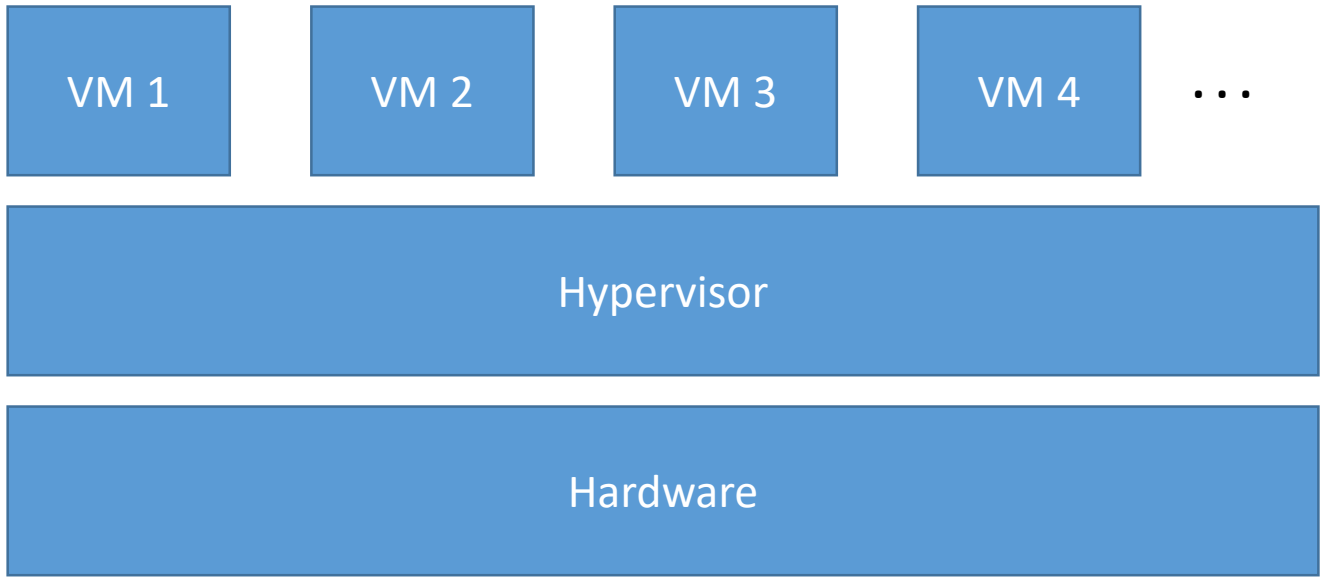
  See Section 24.6.9 for details regarding how these bitmaps are identified.
- **RDPMC.** The RDPMC instruction causes a VM exit if the "RDPMC exiting" VM-execution control is 1.
- **RDRAND.** The RDRAND instruction causes a VM exit if the "RDRAND exiting" VM-execution control is 1.
- **RDSEED.** The RDSEED instruction causes a VM exit if the "RDSEED exiting" VM-execution control is 1.
- **RDTSC.** The RDTSC instruction causes a VM exit if the "RDTSC exiting" VM-execution control is 1.
- **RDTSCP.** The RDTSCP instruction causes a VM exit if the "RDTSC exiting" and "enable RDTSCP" VM-execution controls are both 1.

...

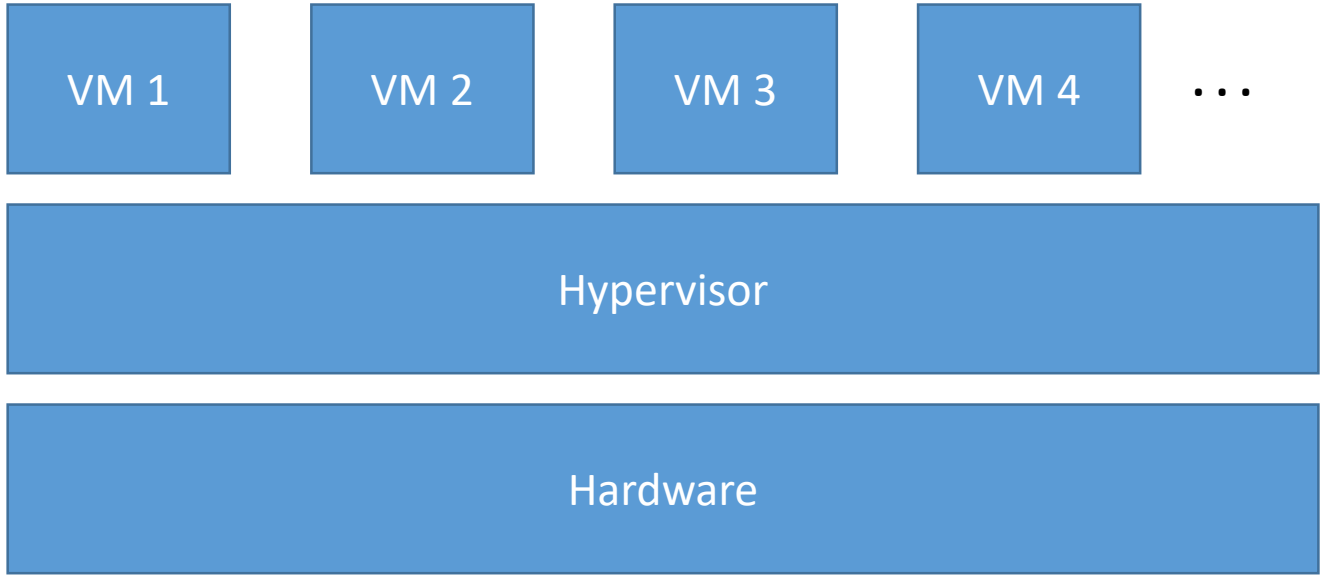
Virtualização da CPU – Exemplo 2/2

- Demo VMCALL

Virtualização de Memória com Intel EPT (Extended Page-Table) – Visão de Alto Nível 1/4

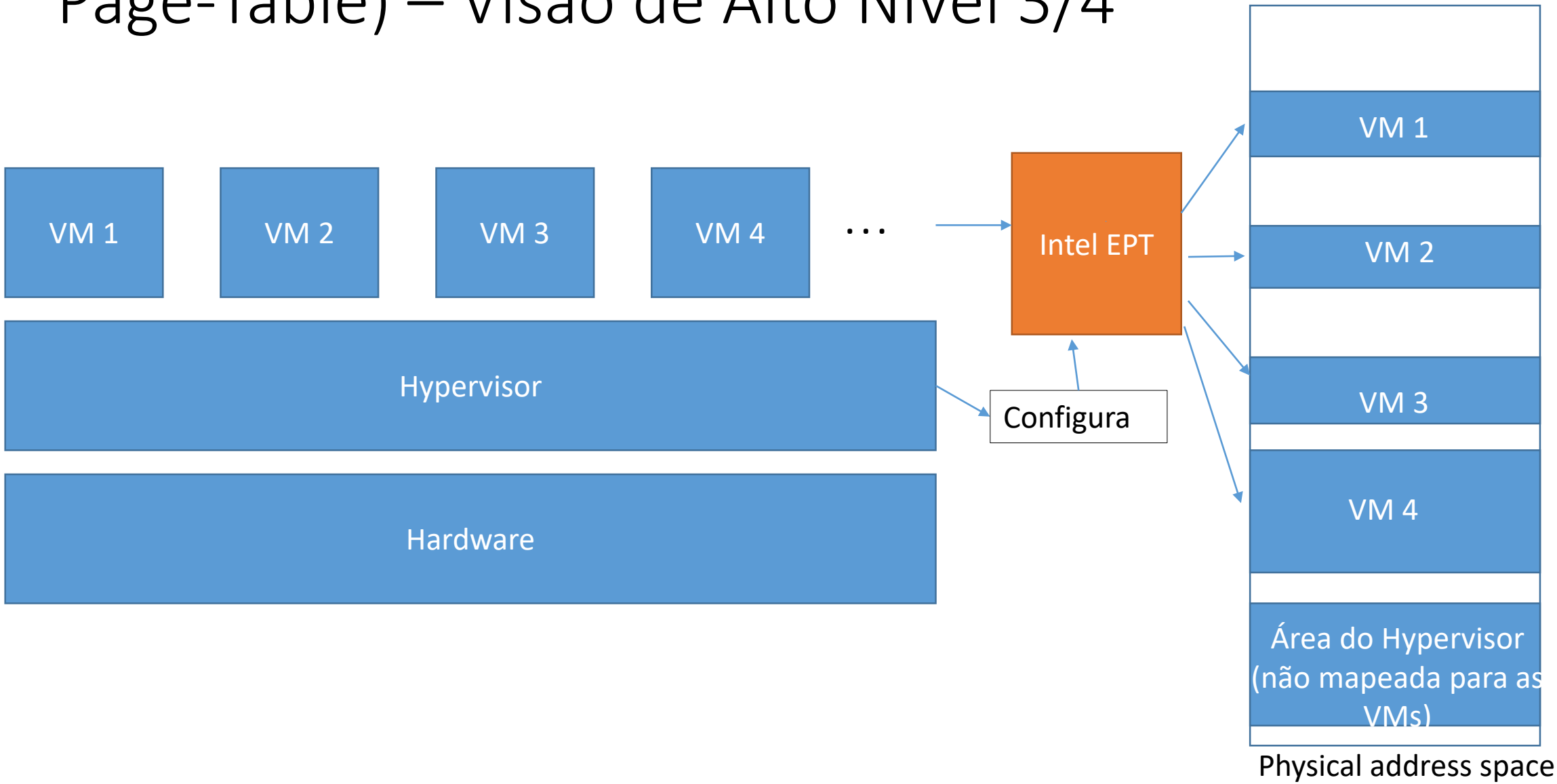


Virtualização de Memória com Intel EPT (Extended Page-Table) – Visão de Alto Nível 2/4



VM	Guest Physical Address (GPA)	System Physical Address (SPA)	Attributes
VM 1	0x0	0x12345678	Read-Only
VM 1	0xdeadbeef	Not mapped	-

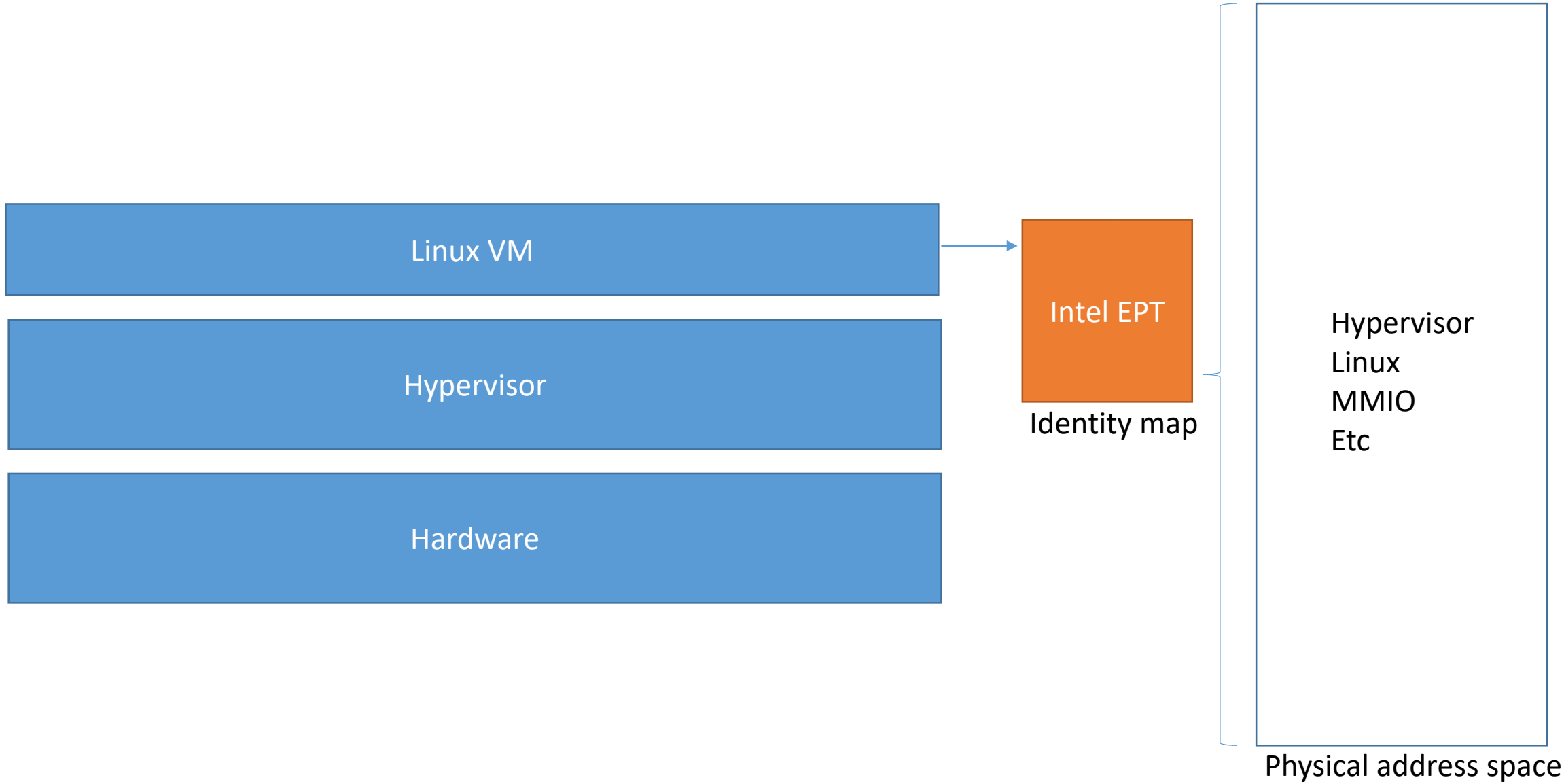
Virtualização de Memória com Intel EPT (Extended Page-Table) – Visão de Alto Nível 3/4



Virtualização de Memória com Intel EPT (Extended Page-Table) – Visão de Alto Nível 4/4

- Demo EPT Pointer

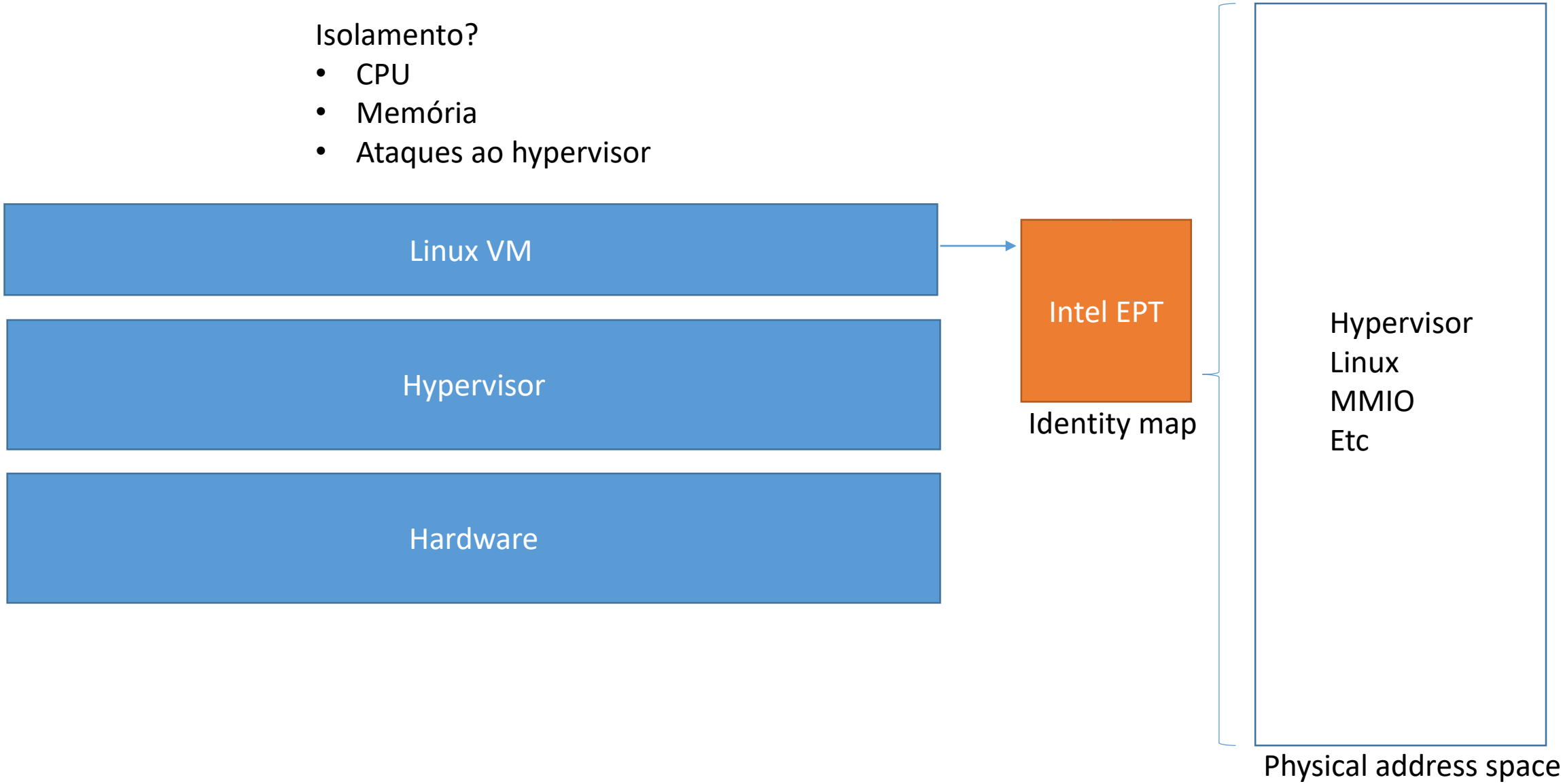
Ambiente 1/3


Linux VM
Hypervisor
Hardware
Intel EPT
Identity map
Hypervisor
Linux
MMIO
Etc
Physical address space

Ambiente 2/3

Isolamento?

- CPU
- Memória
- Ataques ao hypervisor



Ambiente 3/3

- Demo Identity Map + 4Kb pages

Agenda

- Conceitos de virtualização
- Abusando da virtualização
- Conclusões

RDRAND Backdoor 1/3

RDRAND—Read Random Number

Opcode*/ Instruction	Op/ En	64/32 bit Mode Support	CPUID Feature Flag	Description
0F C7 /6 RDRAND r16	M	V/V	RDRAND	Read a 16-bit random number and store in the destination register.
0F C7 /6 RDRAND r32	M	V/V	RDRAND	Read a 32-bit random number and store in the destination register.
REX.W + 0F C7 /6 RDRAND r64	M	V/I	RDRAND	Read a 64-bit random number and store in the destination register.

Instruction Operand Encoding

Op/En	Operand 1	Operand 2	Operand 3	Operand 4
M	ModRM:r/m (w)	NA	NA	NA

Description

Loads a hardware generated random value and store it in the destination register. The size of the random value is determined by the destination register size and operating mode. The Carry Flag indicates whether a random value is available at the time the instruction is executed. CF=1 indicates that the data in the destination is valid. Otherwise CF=0 and the data in the destination operand will be returned as zeros for the specified width. All other flags are forced to 0 in either situation. Software must check the state of CF=1 for determining if a valid random value has been returned, otherwise it is expected to loop and retry execution of RDRAND (see *Intel® 64 and IA-32 Architectures Software Developer's Manual, Volume 1*, Section 7.3.17, "Random Number Generator Instructions").

This instruction is available at all privilege levels.

In 64-bit mode, the instruction's default operation size is 32 bits. Using a REX prefix in the form of REX.B permits access to additional registers (R8-R15). Using a REX prefix in the form of REX.W promotes operation to 64 bit operands. See the summary chart at the beginning of this section for encoding data and limits.

RDRAND Backdoor 2/3

Table 24-7. Definitions of Secondary Processor-Based VM-Execution Controls

Bit Position(s)	Name	Description
0	Virtualize APIC accesses	If this control is 1, the logical processor treats specially accesses to the page with the APIC-access address. See Section 29.4.
1	Enable EPT	If this control is 1, extended page tables (EPT) are enabled. See Section 28.2.
2	Descriptor-table exiting	This control determines whether executions of LGDT, LIDT, LLDT, LTR, SGDT, SIDT, SLDT, and STR cause VM exits.
3	Enable RDTSCP	If this control is 0, any execution of RDTSCP causes an invalid-opcode exception (#UD).
4	Virtualize x2APIC mode	If this control is 1, the logical processor treats specially RDMSR and WRMSR to APIC MSRs (in the range 800H–8FFH). See Section 29.5.
5	Enable VPID	If this control is 1, cached translations of linear addresses are associated with a virtual-processor identifier (VPID). See Section 28.1.
6	WBINVD exiting	This control determines whether executions of WBINVD cause VM exits.
7	Unrestricted guest	This control determines whether guest software may run in unpaged protected mode or in real-address mode.
8	APIC-register virtualization	If this control is 1, the logical processor virtualizes certain APIC accesses. See Section 29.4 and Section 29.5.
9	Virtual-interrupt delivery	This controls enables the evaluation and delivery of pending virtual interrupts as well as the emulation of writes to the APIC registers that control interrupt prioritization.
11	RDRAND exiting	This control determines whether executions of RDRAND cause VM exits.
13	Enable VM functions	Setting this control to 1 enables use of the VMFUNC instruction in VMX non-root operation. See Section 25.5.5.
14	VMCS shadowing	If this control is 1, executions of VMREAD and VMWRITE in VMX non-root operation may access a shadow VMCS (instead of causing VM exits). See Section 24.10 and Section 30.3.
15	Enable ENCLS exiting	If this control is 1, executions of ENCLS consult the ENCLS-exiting bitmap to determine whether the instruction causes a VM exit. See Section 24.6.16 and Section 25.1.3.
16	RDSEED exiting	This control determines whether executions of RDSEED cause VM exits.
17	Enable PML	If this control is 1, an access to a guest-physical address that sets an EPT dirty bit first adds an entry to the page-modification log. See Section 28.2.5.
18	EPT-violation #VE	If this control is 1, EPT violations may cause virtualization exceptions (#VE) instead of VM exits. See Section 25.5.6.
19	Conceal VMX from PT	If this control is 1, Intel Processor Trace suppresses from PIPs an indication that the processor was in VMX non-root operation and omits a VMCS packet from any PSB+ produced in VMX non-root operation (see Chapter 35).
20	Enable XSAVES/XRSTORS	If this control is 0, any execution of XSAVES or XRSTORS causes a #UD.
22	Mode-based execute control for EPT	If this control is 1, EPT execute permissions are based on whether the linear address being accessed is supervisor mode or user mode. See Chapter 28.
25	Use TSC scaling	This control determines whether executions of RDTSC, executions of RDTSCP, and executions of RDMSR that read from the IA32_TIME_STAMP_COUNTER MSR return a value modified by the TSC multiplier field (see Section 24.6.5 and Section 25.3).

RDRAND Backdoor 3/3

- Demo

RDSEED Backdoor 1/3

RDSEED—Read Random SEED

Opcode/ Instruction	Op/ En	64/32 bit Mode Support	CPUID Feature Flag	Description
0F C7 /7 RDSEED r16	M	V/V	RDSEED	Read a 16-bit NIST SP800-90B & C compliant random value and store in the destination register.
0F C7 /7 RDSEED r32	M	V/V	RDSEED	Read a 32-bit NIST SP800-90B & C compliant random value and store in the destination register.
REX.W + 0F C7 /7 RDSEED r64	M	V/I	RDSEED	Read a 64-bit NIST SP800-90B & C compliant random value and store in the destination register.

Instruction Operand Encoding

Op/En	Operand 1	Operand 2	Operand 3	Operand 4
M	ModRM:r/m (w)	NA	NA	NA

Description

Loads a hardware generated random value and store it in the destination register. The random value is generated from an Enhanced NRBG (Non Deterministic Random Bit Generator) that is compliant to NIST SP800-90B and NIST SP800-90C in the XOR construction mode. The size of the random value is determined by the destination register size and operating mode. The Carry Flag indicates whether a random value is available at the time the instruction is executed. CF=1 indicates that the data in the destination is valid. Otherwise CF=0 and the data in the destination operand will be returned as zeros for the specified width. All other flags are forced to 0 in either situation. Software must check the state of CF=1 for determining if a valid random seed value has been returned, otherwise it is expected to loop and retry execution of RDSEED (see Section 1.2).

The RDSEED instruction is available at all privilege levels. The RDSEED instruction executes normally either inside or outside a transaction region.

In 64-bit mode, the instruction's default operation size is 32 bits. Using a REX prefix in the form of REX.B permits access to additional registers (R8-R15). Using a REX prefix in the form of REX.W promotes operation to 64 bit operands. See the summary chart at the beginning of this section for encoding data and limits.

RDSEED Backdoor 2/3

Table 24-7. Definitions of Secondary Processor-Based VM-Execution Controls

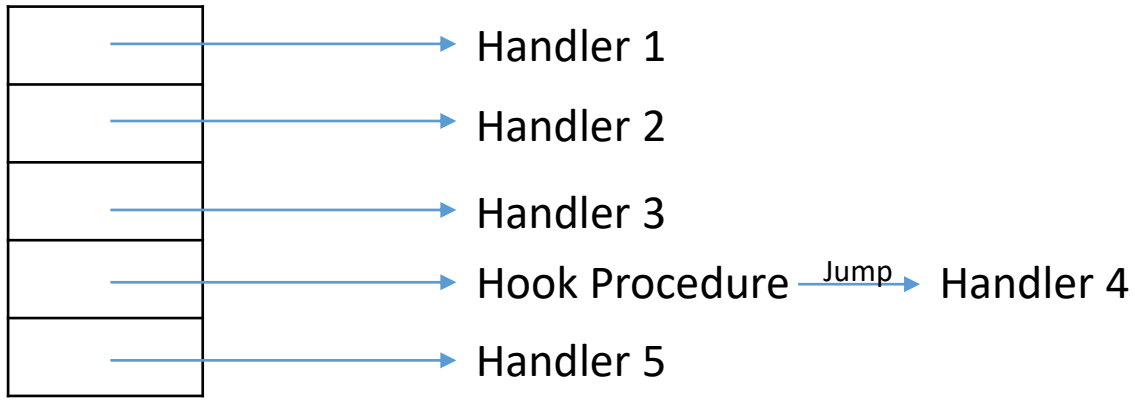
Bit Position(s)	Name	Description
0	Virtualize APIC accesses	If this control is 1, the logical processor treats specially accesses to the page with the APIC-access address. See Section 29.4.
1	Enable EPT	If this control is 1, extended page tables (EPT) are enabled. See Section 28.2.
2	Descriptor-table exiting	This control determines whether executions of LGDT, LIDT, LLDT, LTR, SGDT, SIDT, SLDT, and STR cause VM exits.
3	Enable RDTSCP	If this control is 0, any execution of RDTSCP causes an invalid-opcode exception (#UD).
4	Virtualize x2APIC mode	If this control is 1, the logical processor treats specially RDMSR and WRMSR to APIC MSRs (in the range 800H–8FFH). See Section 29.5.
5	Enable VPID	If this control is 1, cached translations of linear addresses are associated with a virtual-processor identifier (VPID). See Section 28.1.
6	WBINVD exiting	This control determines whether executions of WBINVD cause VM exits.
7	Unrestricted guest	This control determines whether guest software may run in unpaged protected mode or in real-address mode.
8	APIC-register virtualization	If this control is 1, the logical processor virtualizes certain APIC accesses. See Section 29.4 and Section 29.5.
9	Virtual-interrupt delivery	This controls enables the evaluation and delivery of pending virtual interrupts as well as the emulation of writes to the APIC registers that control interrupt prioritization.
10	PAUSE-loop exiting	This control determines whether a series of executions of PAUSE can cause a VM exit (see Section 24.6.13 and Section 25.1.3).
11	RDRAND exiting	This control determines whether executions of RDRAND cause VM exits.
12	Enable INVPCID	If this control is 0, any execution of INVPCID causes a #UD.
13	Enable VM functions	Setting this control to 1 enables use of the VMFUNC instruction in VMX non-root operation. See Section 25.5.5.
14	VMCS shadowing	If this control is 1, executions of VMREAD and VMWRITE in VMX non-root operation may access a shadow VMCS (instead of causing VM exits). See Section 24.10 and Section 30.3.
15	Enable ENCLS	If this control is 1, executions of ENCLS consult the ENCLS-exiting bitmap to determine whether
16	RDSEED exiting	This control determines whether executions of RDSEED cause VM exits.
17	Enable PML	If this control is 1, an access to a guest-physical address that sets an EPT dirty bit first adds an entry to the page-modification log. See Section 28.2.5.
18	EPT-violation #VE	If this control is 1, EPT violations may cause virtualization exceptions (#VE) instead of VM exits. See Section 25.5.6.
19	Conceal VMX from PT	If this control is 1, Intel Processor Trace suppresses from PIPs an indication that the processor was in VMX non-root operation and omits a VMCS packet from any PSB+ produced in VMX non-root operation (see Chapter 35).
20	Enable XSAVES/XRSTORS	If this control is 0, any execution of XSAVES or XRSTORS causes a #UD.
22	Mode-based execute control for EPT	If this control is 1, EPT execute permissions are based on whether the linear address being accessed is supervisor mode or user mode. See Chapter 28.
25	Use TSC scaling	This control determines whether executions of RDTSC, executions of RDTSCP, and executions of RDMSR that read from the IA32_TIME_STAMP_COUNTER MSR return a value modified by the TSC multiplier field (see Section 24.6.5 and Section 25.3).

RDSEED Backdoor 3/3

- Demo

Escondendo sys_call_table Hooks 1/5

- sys_call_table
  - Tabela em memória com ponteiros para handlers das system calls
- sys_call_table hook



Escondendo sys_call_table Hooks 2/5

- Uma forma para detectar esses hooks:
  - Listar as entradas da sys_call_table para encontrar inconsistências
- Demo detectando sys_call_table hooks

Escondendo sys_call_table Hooks 3/5

Table 28-6. Format of an EPT Page-Table Entry that Maps a 4-KByte Page

Bit Position(s)	Contents
0	Read access; indicates whether reads are allowed from the 4-KByte page referenced by this entry
1	Write access; indicates whether writes are allowed from the 4-KByte page referenced by this entry
2	If the "mode-based execute control for EPT" VM-execution control is 0, execute access; indicates whether instruction fetches are allowed from the 4-KByte page controlled by this entry If that control is 1, execute access for supervisor-mode linear addresses; indicates whether instruction fetches are allowed from supervisor-mode linear addresses in the 4-KByte page controlled by this entry
5:3	EPT memory type for this 4-KByte page (see Section 28.2.6)
6	Ignore PAT memory type for this 4-KByte page (see Section 28.2.6)
7	Ignored
8	If bit 6 of EPTP is 1, accessed flag for EPT; indicates whether software has accessed the 4-KByte page referenced by this entry (see Section 28.2.4). Ignored if bit 6 of EPTP is 0
9	If bit 6 of EPTP is 1, dirty flag for EPT; indicates whether software has written to the 4-KByte page referenced by this entry (see Section 28.2.4). Ignored if bit 6 of EPTP is 0
10	Execute access for user-mode linear addresses. If the "mode-based execute control for EPT" VM-execution control is 1, indicates whether instruction fetches are allowed from user-mode linear addresses in the 4-KByte page controlled by this entry. If that control is 0, this bit is ignored.
11	Ignored
(N–1):12	Physical address of the 4-KByte page referenced by this entry[1]
51:N	Reserved (must be 0)
62:52	Ignored
63	Suppress #VE. If the "EPT-violation #VE" VM-execution control is 1, EPT violations caused by accesses to this page are convertible to virtualization exceptions only if this bit is 0 (see Section 25.5.6.1). If "EPT-violation #VE" VM-execution control is 0, this bit is ignored.

Escondendo sys_call_table Hooks 4/5

Table 24-6. Definitions of Primary Processor-Based VM-Execution Controls (Contd.)

Bit Position(s)	Name	Description
15	CR3-load exiting	In conjunction with the CR3-target controls (see Section 24.6.7), this control determines whether executions of MOV to CR3 cause VM exits. See Section 25.1.3. The first processors to support the virtual-machine extensions supported only the 1-setting of this control.
16	CR3-store exiting	This control determines whether executions of MOV from CR3 cause VM exits. The first processors to support the virtual-machine extensions supported only the 1-setting of this control.
19	CR8-load exiting	This control determines whether executions of MOV to CR8 cause VM exits.
20	CR8-store exiting	This control determines whether executions of MOV from CR8 cause VM exits.
21	Use TPR shadow	Setting this control to 1 enables TPR virtualization and other APIC-virtualization features. See Chapter 29.
22	NMI-window exiting	If this control is 1, a VM exit occurs at the beginning of any instruction if there is no virtual-NMI blocking (see Section 24.4.2).
23	MOV-DR exiting	This control determines whether executions of MOV DR cause VM exits.
24	Unconditional I/O exiting	This control determines whether executions of I/O instructions (IN, INS/INSB/INSW/INSD, OUT, and OUTS/OUTSB/OUTSW/OUTSD) cause VM exits.
25	Use I/O bitmaps	This control determines whether I/O bitmaps are used to restrict executions of I/O instructions (see Section 24.6.4 and Section 25.1.3). For this control, "0" means "do not use I/O bitmaps" and "1" means "use I/O bitmaps." If the I/O bitmaps are used, the setting of the "unconditional I/O exiting" control is ignored.
27	Monitor trap flag	If this control is 1, the monitor trap flag debugging feature is enabled. See Section 25.5.2.
28	Use MSR bitmaps	This control determines whether MSR bitmaps are used to control execution of the RDMSR and WRMSR instructions (see Section 24.6.9 and Section 25.1.3). For this control, "0" means "do not use MSR bitmaps" and "1" means "use MSR bitmaps." If the MSR bitmaps are not used, all executions of the RDMSR and WRMSR instructions cause VM exits.
29	MONITOR exiting	This control determines whether executions of MONITOR cause VM exits.
30	PAUSE exiting	This control determines whether executions of PAUSE cause VM exits.
31	Activate secondary controls	This control determines whether the secondary processor-based VM-execution controls are used. If this control is 0, the logical processor operates as if all the secondary processor-based VM-execution controls were also 0.

25.5.2 Monitor Trap Flag

The **monitor trap flag** is a debugging feature that causes VM exits to occur on certain instruction boundaries in VMX non-root operation. Such VM exits are called **MTF VM exits**. An MTF VM exit may occur on an instruction boundary in VMX non-root operation as follows:

…

Escondendo sys_call_table Hooks 5/5

- Estratégia para esconder hooks (baseada nas ideias de http://www.phrack.org/issues/69/15.html)
  1. Clonar a página EPT da sys_call_table antes do hook
     - Isso é que o sys_call_table legítimo deve ser
  2. Hookar sys_call_table
  3. Setar permissão de execução na página EPT da sys_call_table
     - Execuções funcionarão normalmente
  4. Desabilitar permissões de leitura e escrita na página EPT da sys_call_table
     - Leituras e escritas gerarão VM exit
  5. No VM exit por causa de #4:
     a. Trocar a página EPT da sys_call_table pela clonada
     b. Setar Monitor Trap Flag
        - VM exit será gerado quando a instrução terminar
  6. No VM exit por causa de #5-b, voltar a página EPT da sys_call_table com os hooks

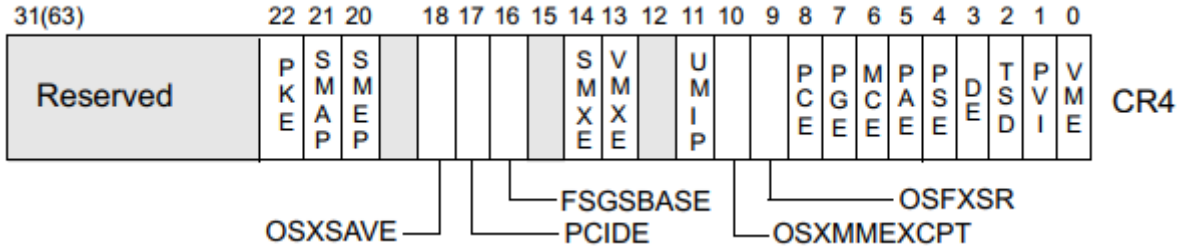
- Demo

Algumas Ideias sobre Comunicação com o Hypervisor

- VMCALL
  - Facilmente detectável, então não é uma opção muito boa nesse sentido
- Instruções que causam VM exit (condicional ou incondicional) são uma opção melhor
  - Determinados valores em registradores podem ser utilizados para diferenciar a utilização legítima da instrução do método de comunicação
  - Por exemplo, CPUID
    - Pode ser utilizada em qualquer nível de privilégio
    - Possui uso legítimo que pode ser utilizado para mascarar a comunicação com o hypervisor.
      - Por exemplo: SE recurso x está disponível ENTÃO chamar foo()
    - Registradores são utilizados como parâmetro para a CPUID, e são bem documentados
      - Então, é possível encontrar valores não conflitantes para realizar a comunicação com o hypervisor
- Demo

Algumas Ideias para Esconder a Virtualização 1/2

- Ocultar rastros do VMX na CPU. Por exemplo, CR4.VMXE:



CR4.VMXE

VMX-Enable Bit (bit 13 of CR4) — Enables VMX operation when set. See Chapter 23, "Introduction to Virtual Machine Extensions."

- Demo

Algumas Ideias para Esconder a Virtualização 2/2

- Ocultar determinados comportamentos, como TSC
  - Não é fácil mascarar completamente o TSC:
    - Aplicacões legacy
    - Existência de diversos métodos de medida
    - Diferentes comportamentos para a mesma instrução (por exemplo, CPUID)
    - Etc

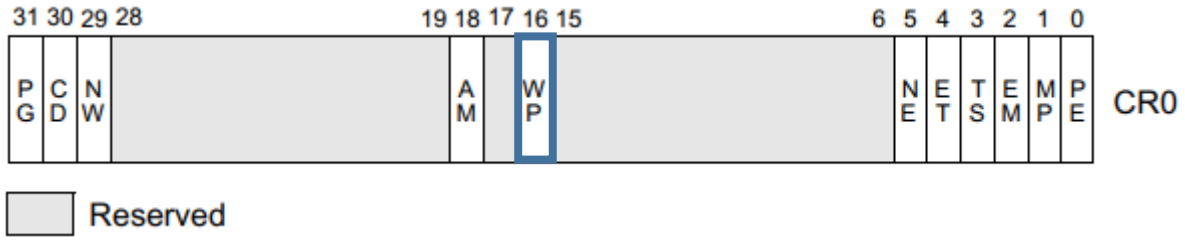
Table 24-6. Definitions of Primary Processor-Based VM-Execution Controls

Bit Position(s)	Name	Description
2	Interrupt-window exiting	If this control is 1, a VM exit occurs at the beginning of any instruction if RFLAGS.IF = 1 and there are no other blocking of interrupts (see Section 24.4.2).
3	Use TSC offsetting	This control determines whether executions of RDTSC, executions of RDTSCP, and executions of RDMSR that read from the IA32_TIME_STAMP_COUNTER MSR return a value modified by the TSC offset field (see Section 24.6.5 and Section 25.3).
7	HLT exiting	This control determines whether executions of HLT cause VM exits.
9	INVLPG exiting	This determines whether executions of INVLPG cause VM exits.
10	MWAIT exiting	This control determines whether executions of MWAIT cause VM exits.
11	RDPMC exiting	This control determines whether executions of RDPMC cause VM exits.
12	RDTSC exiting	This control determines whether executions of RDTSC and RDTSCP cause VM exits.

- Demo

Uma forma de proteger a sys_call_table 1/3

- sys_call_table geralmente reside em páginas read-only



CR0.WP

Write Protect (bit 16 of CR0) — When set, inhibits supervisor-level procedures from writing into read-only pages; when clear, allows supervisor-level procedures to write into read-only pages (regardless of the U/S bit setting; see Section 4.1.3 and Section 4.6). This flag facilitates implementation of the copy-on-write method of creating a new process (forking) used by operating systems such as UNIX.

Uma forma de proteger a sys_call_table 2/3

- **MOV to CR0.** The MOV to CR0 instruction causes a VM exit unless the value of its source operand matches, for the position of each bit set in the CR0 guest/host mask, the corresponding bit in the CR0 read shadow. (If every bit is clear in the CR0 guest/host mask, MOV to CR0 cannot cause a VM exit.)

24.6.6 Guest/Host Masks and Read Shadows for CR0 and CR4

VM-execution control fields include **guest/host masks** and **read shadows** for the CR0 and CR4 registers. These fields control executions of instructions that access those registers (including CLTS, LMSW, MOV CR, and SMSW). They are 64 bits on processors that support Intel 64 architecture and 32 bits on processors that do not.

In general, bits set to 1 in a guest/host mask correspond to bits "owned" by the host:

- Guest attempts to set them (using CLTS, LMSW, or MOV to CR) to values differing from the corresponding bits in the corresponding read shadow cause VM exits.
- Guest reads (using MOV from CR or SMSW) return values for these bits from the corresponding read shadow.

Bits cleared to 0 correspond to bits "owned" by the guest; guest attempts to modify them succeed and guest reads return values for these bits from the control register itself.

See Chapter 27 for details regarding how these fields affect VMX non-root operation.

Uma forma de proteger a sys_call_table 3/3

- Setup:
  1. Setar CR0.WP (bit 16) no guest/host masks para CR0
  2. Setar CR0.WP (bit 16) no read shadows para CR0
- Prevenção:
  - VM exit handler relacionado com a escrita no CR0 pode continuar a execução, simplesmente pulando a instrução que escreveu no CR0
    - A instrução não terá efeito algum
- Detecção:
  - VM exit handler relacionado com a escrita do CR0 pode logar a tentativa de escrita
- Demo

Áreas de Memória Protegidas

- Pode ser necessário proteger determinados segredos como chaves criptográficas até mesmo de ring 0 (VM).
  - Exemplo: Uma abordagem estilo TPM, onde ring3/ring0 podem solicitar operações criptográficas ao hypervisor sem terem acesso às chaves
- Existem diversas opções para implementar isso
  - Por exemplo, não mapear a página com o segredo para a VM e definir um método de comunicação com o hypervisor

Agenda

- Conceitos de virtualização
- Abusando da virtualização
- Conclusões

Conclusões

- O sistema operacional pode ser virtualizado, e então o hypervisor apareceria como uma camada mais privilegiada abaixo dele
- Diversos recursos existem para o hypervisor controlar VMs
  - Malware pode fazer uso deles de diversas formas
  - Software de segurança também pode os utilizar de diversas formas
- Obter um hypervisor totalmemte invisível é um desafio!
  - Mas atingir um nível de invisibilidade “bom o suficiete” de forma que certas técnicas sejam mitigadas é factível
- Tome cuidado em que serviço cloud você está colocando seus sistemas!
  - A empresa é confiável?
  - A empresa está comprometida?

Muito Obrigado!!!

Gabriel Negreira Barbosa
gabriel.negreira.barbosa [arroba] intel.com