The intersection between virtualization and security: a holistic view

Gabriel Negreira Barbosa

ganegrei [at] microsoft [dot] com

Disclaimer

- I don’t speak on behalf of my employer.
  - All the ideas and information presented here are from myself.

Agenda

- Introduction
- Architectural review
- Attack surface discussion
- Leveraging virtualization for security improvement
- Conclusion

Agenda

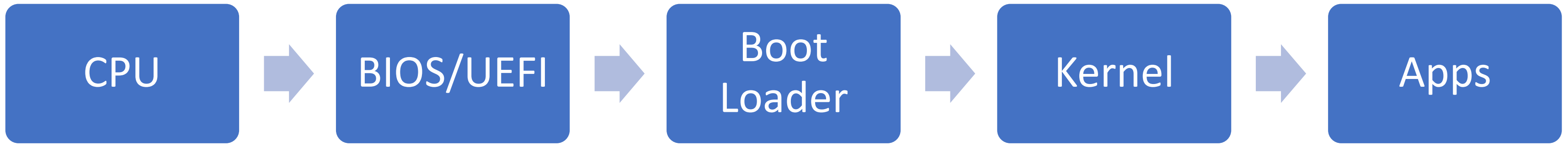
Introduction

- This presentation focuses on:
  - Intel Architecture
  - Bare Metal Hypervisor

Agenda

- Introduction
- Architectural review
- Attack surface discussion
- Leveraging virtualization for security improvement
- Conclusion

Architectural Review – Basic Boot Process

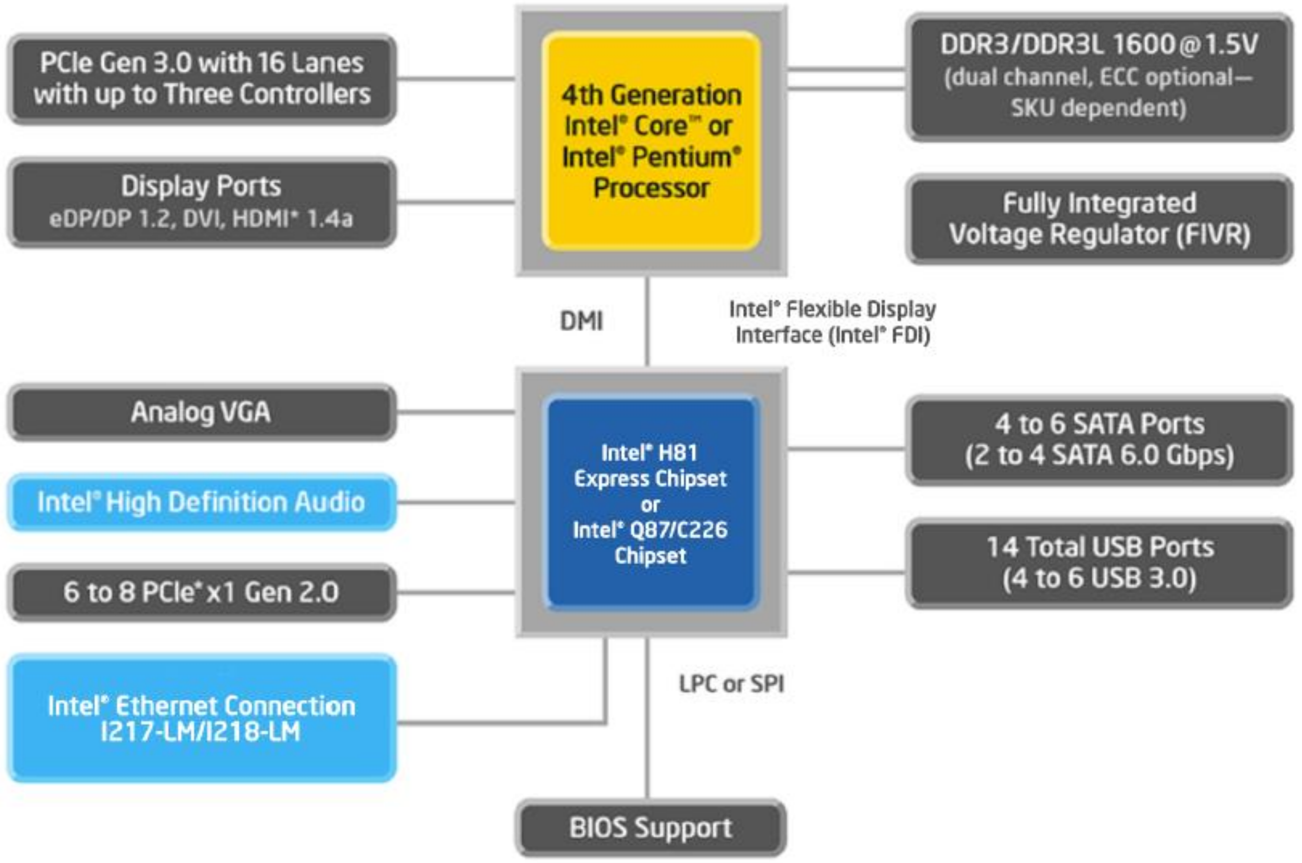


Architectural Review – Boot Process – Attacks Brainstorm

- CPU - - - > > > Apps
    - Are you afraid of this?

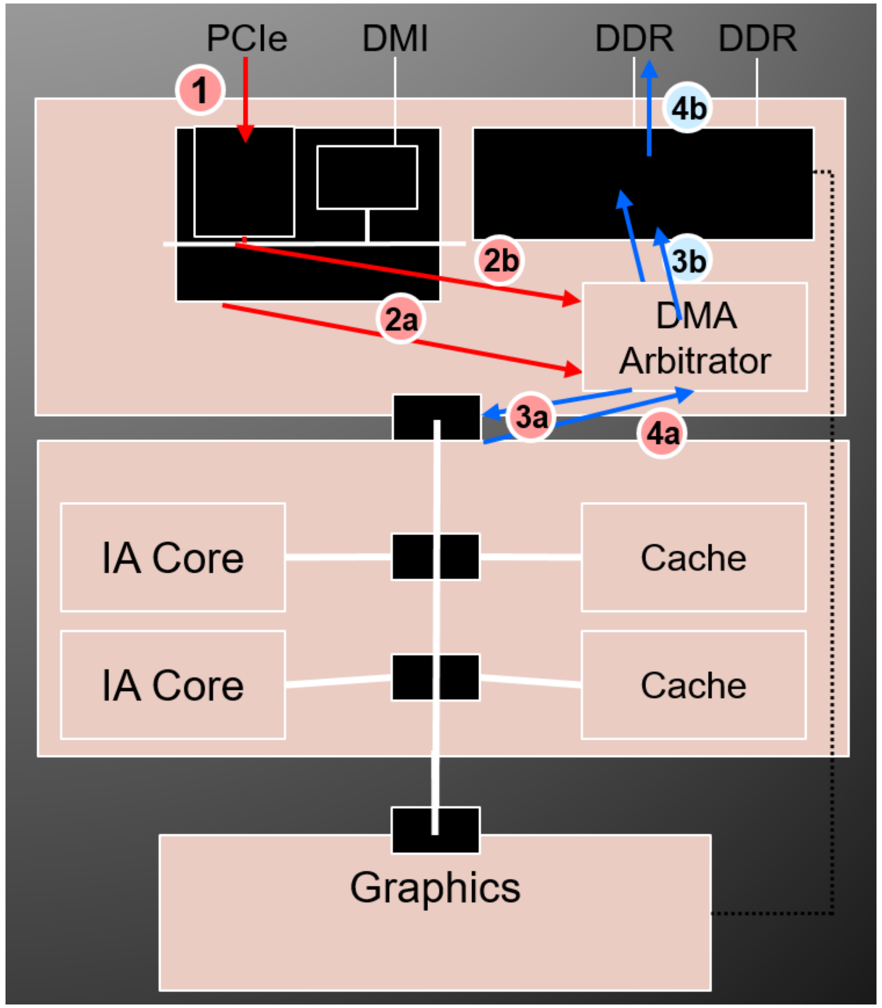
- Apps - - - > > > CPU
    - This sounds scary ☺

Architecture Review


PCIe Gen 3.0 with 16 Lanes with up to Three Controllers
Display Ports eDP/DP 1.2, DVI, HDMI* 1.4a
4th Generation Intel® Core™ or Intel® Pentium® Processor
DDR3/DDR3L 1600@1.5V (dual channel, ECC optional—SKU dependent)
Fully Integrated Voltage Regulator (FIVR)
DMI
Intel® Flexible Display Interface (Intel® FDI)
Analog VGA
Intel® High Definition Audio
6 to 8 PCIe* x1 Gen 2.0
Intel® H81 Express Chipset or Intel® Q87/C226 Chipset
4 to 6 SATA Ports (2 to 4 SATA 6.0 Gbps)
14 Total USB Ports (4 to 6 USB 3.0)
Intel® Ethernet Connection I217-LM/I218-LM
LPC or SPI
BIOS Support

Source: http://www.intel.com/content/www/us/en/intelligent-systems/embedded-systems-training/ia-introduction-basics-paper.html

Architecture Review – DMA



Source: Troopers 2015 – “Modern Platform Supported Rootkits” - https://github.com/rrbranco/Troopers2015

Architectural Review – Devices

- 2 main ways to interact with a device:
  - IO Ports
    - IN/OUT (and similar) instructions
  - MMIO
    - Physical memory accesses decoded to devices

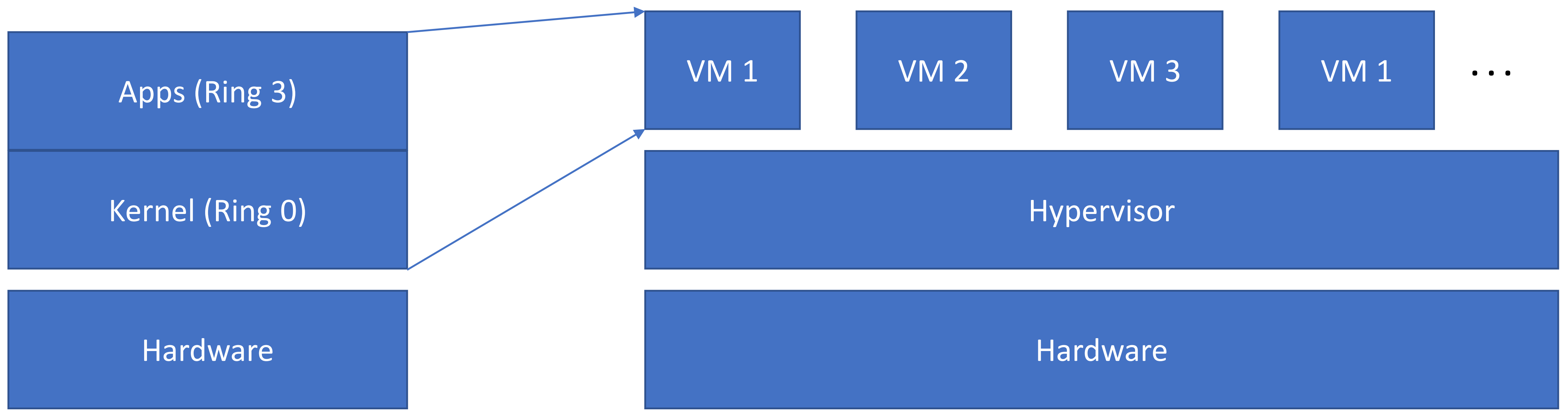
Architectural Review – Other Important Concepts

- MSR
  - The processor provides a variety of machine specific registers that are used to control and report on processor performance. Virtually all MSRs handle **system related functions** and are not accessible to an application program.
  - Examples: 0x3A (IA32_FEATURE_CONTROL), 0x79 (IA32_BIOS_UPDT_TRIG (BIOS_UPDT_TRIG)), 0x1A0 (IA32_MISC_ENABLE), etc.

- SMM
  - SMM is a special-purpose operating mode provided for handling system-wide functions like power management, system hardware control, or proprietary OEM-designed code. It is intended for use only by **system firmware**, not by applications software or general-purpose systems software. The main benefit of SMM is that it offers a distinct and easily isolated processor environment that **operates transparently to the operating system or executive and software applications**.

Source: http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html

Architectural Review - Virtualization



Most modern operating systems

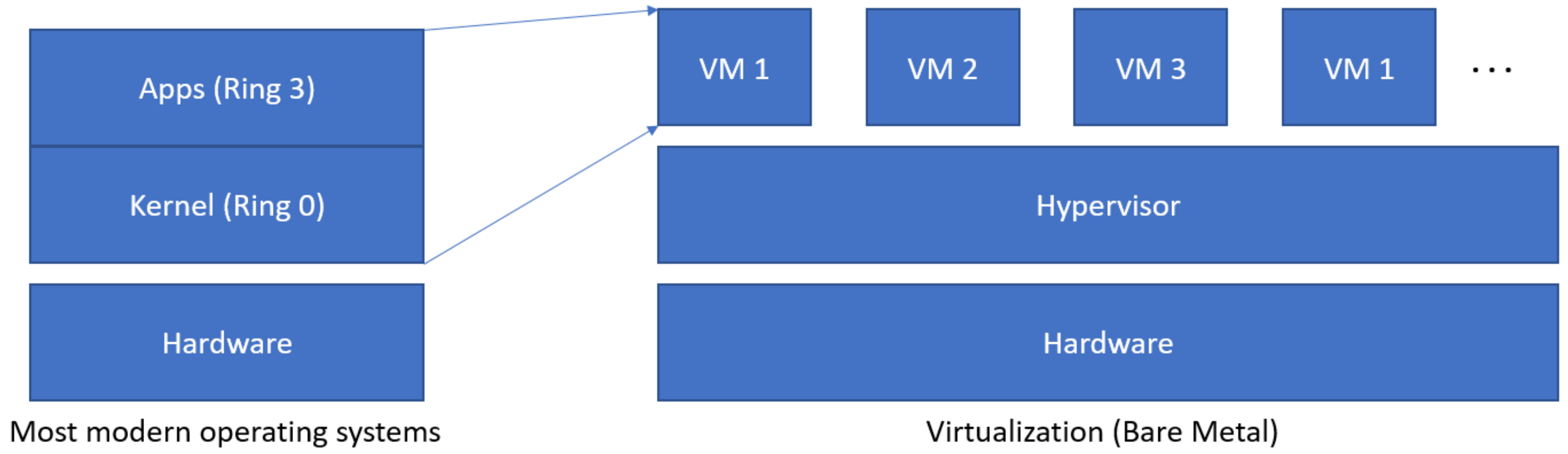
Virtualization (Bare Metal)

Agenda

- Introduction
- Architectural review
- Attack surface discussion
- Leveraging virtualization for security improvement
- Conclusion

Attack Surface – General virtualization attacks brainstorm

- VM to VM
- VM to Host
- VM to Hypervisor
- DoS
- Host to Hypervisor
- BIOS/UEFI/SMM
- Hardware



Attack Surface - Devices

- Mainly 3 ways to assign a device to a VM:
  - Emulation
  - Paravirtualization
  - Direct assignment

Attack Surface – Devices – Emulation

- Devices are emulated by “someone” in the virtualization solution
  - “Someone” = Hypervisor, specific VM, etc
  - Trapped and emulated
    - IO Ports
    - MMIO

Attack Surface – Devices – Emulation

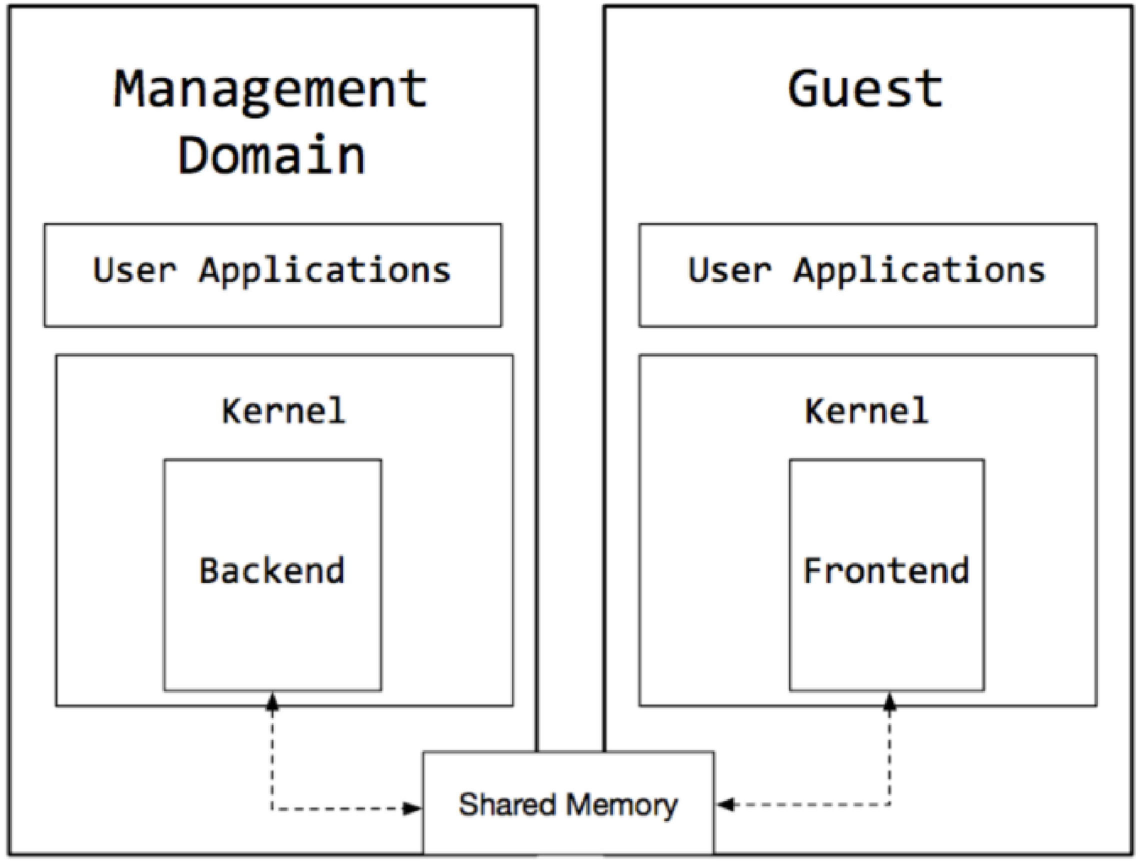
```
--- a/hw/fdc.c
+++ b/hw/fdc.c
@@ -1497,7 +1497,7 @@ static uint32_t fdctrl_read_data(FDCtrl *fdctrl)
 {
     FDrive *cur_drv;
     uint32_t retval = 0;
-    int pos;
+    uint32_t pos;
 
     cur_drv = get_cur_drv(fdctrl);
     fdctrl->dsr &= ~FD_DSR_PWRDOWN;
@@ -1506,8 +1506,8 @@ static uint32_t fdctrl_read_data(FDCtrl *fdctrl)
         return 0;
     }
     pos = fdctrl->data_pos;
+    pos %= FD_SECTOR_LEN;
     if (fdctrl->msr & FD_MSR_NONDMA) {
-        pos %= FD_SECTOR_LEN;
         if (pos == 0) {
             if (fdctrl->data_pos != 0)
                 if (!fdctrl_seek_to_next_sect(fdctrl, cur_drv)) {
@@ -1852,10 +1852,13 @@ static void fdctrl_handle_option(FDCtrl *fdctrl, int direction)
 static void fdctrl_handle_drive_specification_command(FDCtrl *fdctrl, int direction)
 {
     FDrive *cur_drv = get_cur_drv(fdctrl);
+    uint32_t pos;
 
-    if (fdctrl->fifo[fdctrl->data_pos - 1] & 0x80) {
+    pos = fdctrl->data_pos - 1;
+    pos %= FD_SECTOR_LEN;
+    if (fdctrl->fifo[pos] & 0x80) {
         /* Command parameters done */
-        if (fdctrl->fifo[fdctrl->data_pos - 1] & 0x40) {
+        if (fdctrl->fifo[pos] & 0x40) {
             fdctrl->fifo[0] = fdctrl->fifo[1];
             fdctrl->fifo[2] = 0;
             fdctrl->fifo[3] = 0;
```

```
@@ -1955,7 +1958,7 @@ static uint8_t command_to_handler[256];
 static void fdctrl_write_data(FDCtrl *fdctrl, uint32_t value)
 {
     FDrive *cur_drv;
-    int pos;
+    uint32_t pos;
 
     /* Reset mode */
     if (!(fdctrl->dor & FD_DOR_nRESET)) {
@@ -2004,7 +2007,9 @@ static void fdctrl_write_data(FDCtrl *fdctrl, uint32_t value)
     }
 
     FLOPPY_DPRINTF("%s: %02x\n", __func__, value);
-    fdctrl->fifo[fdctrl->data_pos++] = value;
+    pos = fdctrl->data_pos++;
+    pos %= FD_SECTOR_LEN;
+    fdctrl->fifo[pos] = value;
     if (fdctrl->data_pos == fdctrl->data_len) {
         /* We now have all parameters
          * and will be able to treat the command
--
2.1.0
```

- CVE-2015-3456 - VENOM (Virtualized Environment Neglected Operations Manipulation)

http://xenbits.xen.org/xsa/advisory-133.html

Attack Surface – Devices – Paravirtualization

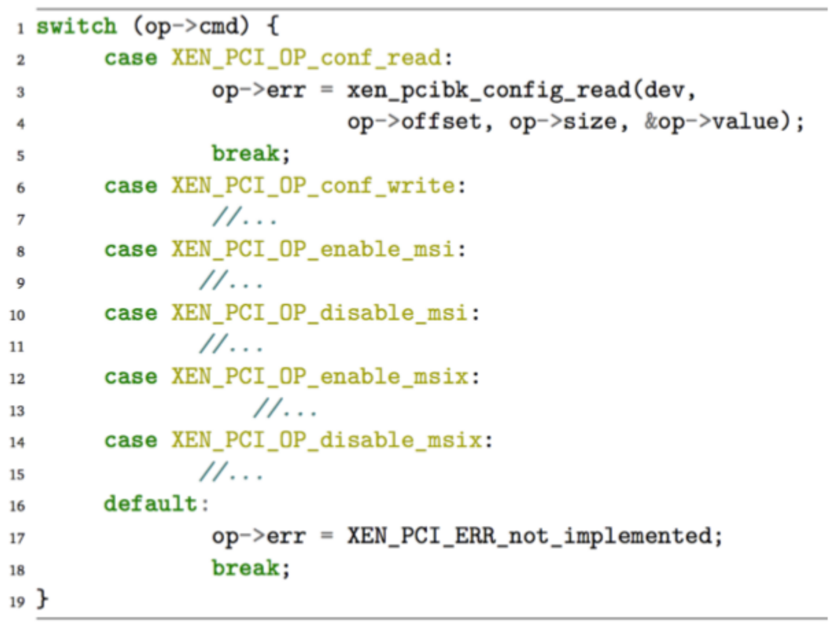


Source: Black Hat 2016 Las Vegas – “Xenpwn – Breaking Paravirtualized Devices” - https://www.blackhat.com/docs/us-16/materials/us-16-Wilhelm-Xenpwn-Breaking-Paravirtualized-Devices.pdf

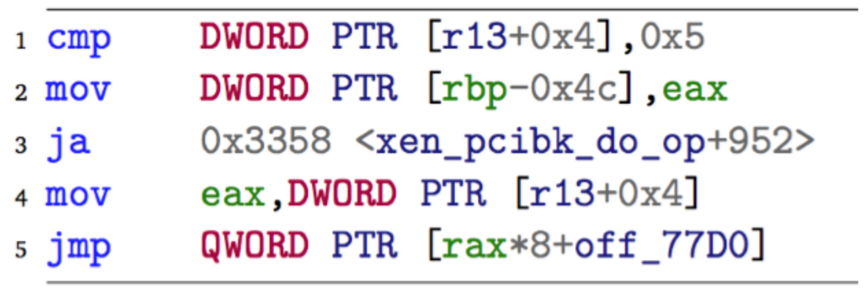
Attack Surface – Devices – Paravirtualization

- XenPwn
  - https://www.blackhat.com/docs/us-16/materials/us-16-Wilhelm-Xenpwn-Breaking-Paravirtualized-Devices-wp.pdf
  - https://www.blackhat.com/docs/us-16/materials/us-16-Wilhelm-Xenpwn-Breaking-Paravirtualized-Devices.pdf

xen-pciback: xen_pcibk_do_op



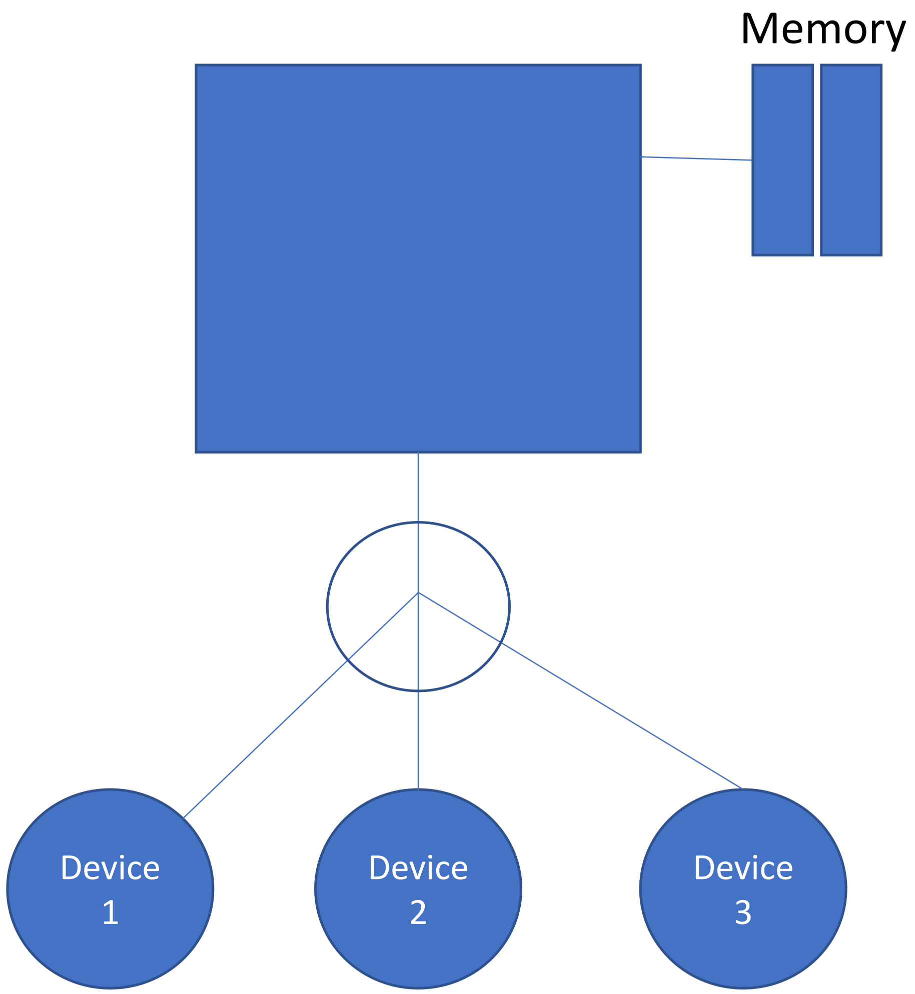
```c
switch (op->cmd) {
    case XEN_PCI_OP_conf_read:
            op->err = xen_pcibk_config_read(dev,
                    op->offset, op->size, &op->value);
            break;
    case XEN_PCI_OP_conf_write:
            //...
    case XEN_PCI_OP_enable_msi:
            //...
    case XEN_PCI_OP_disable_msi:
            //...
    case XEN_PCI_OP_enable_msix:
            //...
    case XEN_PCI_OP_disable_msix:
            //...
    default:
            op->err = XEN_PCI_ERR_not_implemented;
            break;
}
```



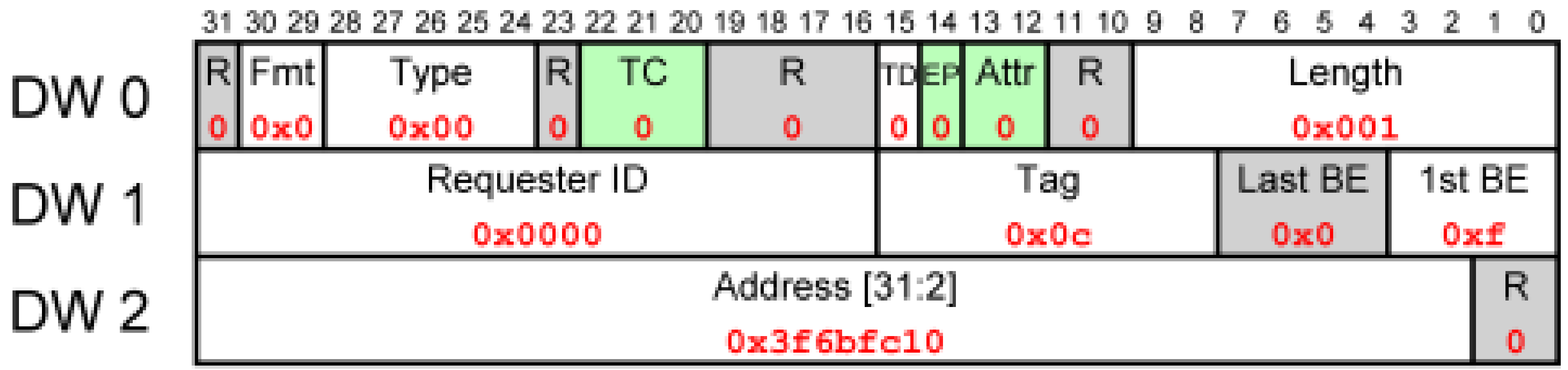
```asm
cmp     DWORD PTR [r13+0x4],0x5
mov     DWORD PTR [rbp-0x4c],eax
ja      0x3358 <xen_pcibk_do_op+952>
mov     eax,DWORD PTR [r13+0x4]
jmp     QWORD PTR [rax*8+off_77D0]
```

Source: Black Hat 2016 Las Vegas – "Xenpwn – Breaking Paravirtualized Devices" - https://www.blackhat.com/docs/us-16/materials/us-16-Wilhelm-Xenpwn-Breaking-Paravirtualized-Devices.pdf

Attack Surface – Devices – Direct Assignment



Attack Surface – Devices – Direct Assignment



TLP read packet

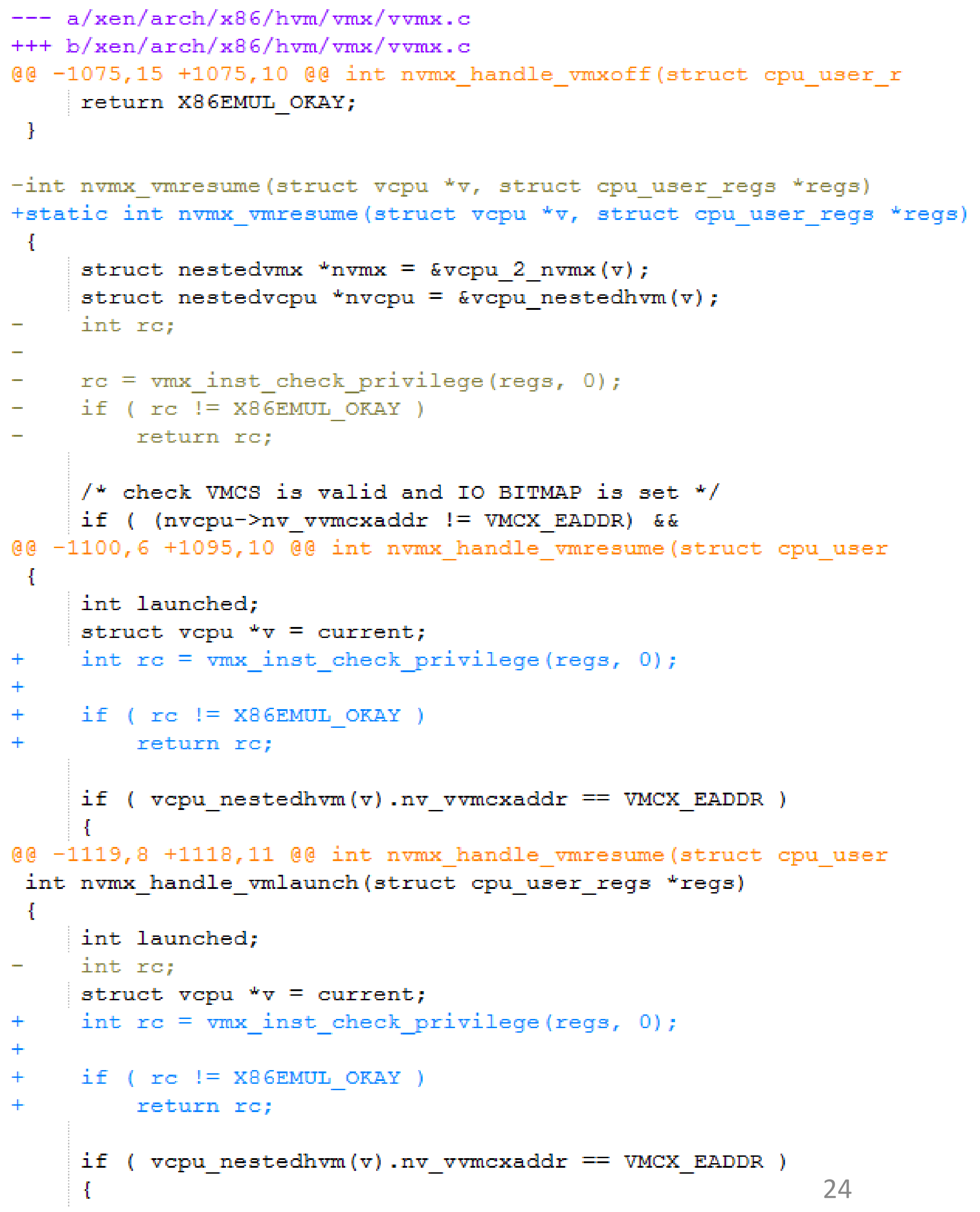
Source: http://xillybus.com/tutorials/pci-express-tlp-pcie-primer-tutorial-guide-1

Attack Surface – Devices – Direct Assignment – Brainstorm

- P2P
- ACS (Access Control System)
- ATS (Address Translation Service)
- “IOMMU”

Attack Surface – “VMEXIT handling”

- https://xenbits.xen.org/xsa/advisory-75.html



```
--- a/xen/arch/x86/hvm/vmx/vvmx.c
+++ b/xen/arch/x86/hvm/vmx/vvmx.c
@@ -1075,15 +1075,10 @@ int nvmx_handle_vmxoff(struct cpu_user_r
     return X86EMUL_OKAY;
 }

-int nvmx_vmresume(struct vcpu *v, struct cpu_user_regs *regs)
+static int nvmx_vmresume(struct vcpu *v, struct cpu_user_regs *regs)
 {
     struct nestedvmx *nvmx = &vcpu_2_nvmx(v);
     struct nestedvcpu *nvcpu = &vcpu_nestedhvm(v);
-    int rc;
-
-    rc = vmx_inst_check_privilege(regs, 0);
-    if ( rc != X86EMUL_OKAY )
-        return rc;

     /* check VMCS is valid and IO BITMAP is set */
     if ( (nvcpu->nv_vvmcxaddr != VMCX_EADDR) &&
@@ -1100,6 +1095,10 @@ int nvmx_handle_vmresume(struct cpu_user
 {
     int launched;
     struct vcpu *v = current;
+    int rc = vmx_inst_check_privilege(regs, 0);
+
+    if ( rc != X86EMUL_OKAY )
+        return rc;

     if ( vcpu_nestedhvm(v).nv_vvmcxaddr == VMCX_EADDR )
     {
@@ -1119,8 +1118,11 @@ int nvmx_handle_vmresume(struct cpu_user
 int nvmx_handle_vmlaunch(struct cpu_user_regs *regs)
 {
     int launched;
-    int rc;
     struct vcpu *v = current;
+    int rc = vmx_inst_check_privilege(regs, 0);
+
+    if ( rc != X86EMUL_OKAY )
+        return rc;

     if ( vcpu_nestedhvm(v).nv_vvmcxaddr == VMCX_EADDR )
     {
```

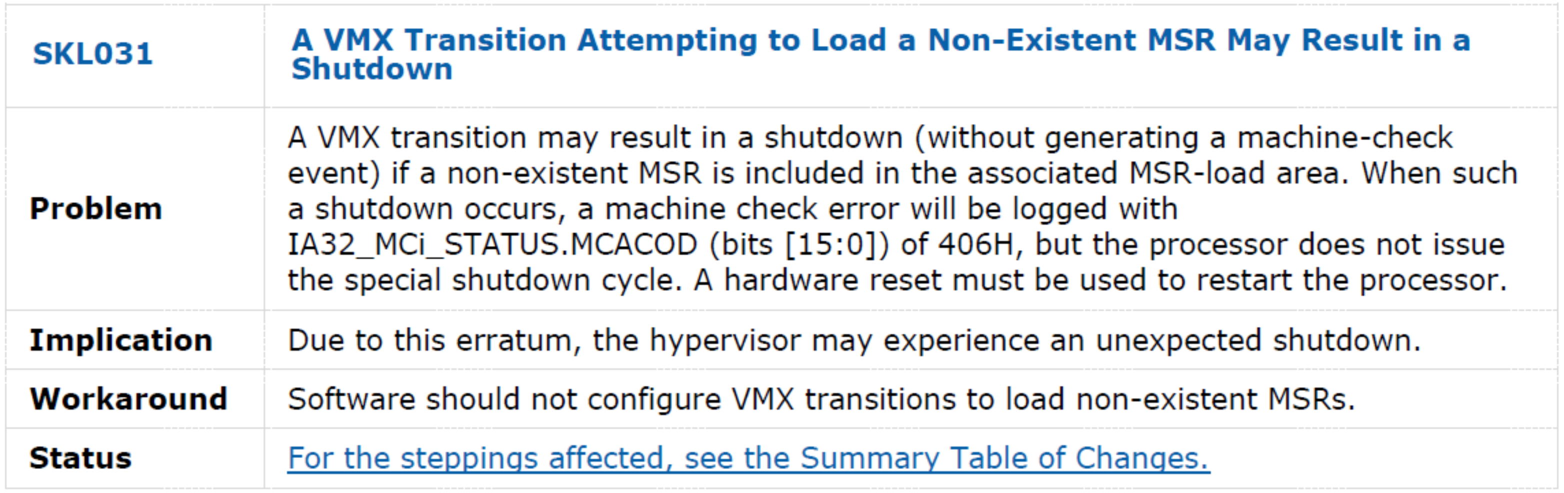
Attack Surface – Hypercall

- https://xenbits.xen.org/xsa/advisory-122.html

```
--- a/xen/common/kernel.c
+++ b/xen/common/kernel.c
@@ -240,6 +240,8 @@ DO(xen_version)(int cmd, XEN_GUEST_HANDL
     case XENVER_extraversion:
     {
         xen_extraversion_t extraversion;
+
+        memset(extraversion, 0, sizeof(extraversion));
         safe_strcpy(extraversion, xen_extra_version());
         if ( copy_to_guest(arg, extraversion, ARRAY_SIZE(extraversion)) )
             return -EFAULT;
@@ -249,6 +251,8 @@ DO(xen_version)(int cmd, XEN_GUEST_HANDL
     case XENVER_compile_info:
     {
         struct xen_compile_info info;
+
+        memset(&info, 0, sizeof(info));
         safe_strcpy(info.compiler,       xen_compiler());
         safe_strcpy(info.compile_by,     xen_compile_by());
         safe_strcpy(info.compile_domain, xen_compile_domain());
@@ -284,6 +288,8 @@ DO(xen_version)(int cmd, XEN_GUEST_HANDL
     case XENVER_changeset:
     {
         xen_changeset_info_t chgset;
+
+        memset(chgset, 0, sizeof(chgset));
         safe_strcpy(chgset, xen_changeset());
         if ( copy_to_guest(arg, chgset, ARRAY_SIZE(chgset)) )
             return -EFAULT;
```

Attack Surface – Hardware Virtualization Support

- Bugs on the underlying infrastructure

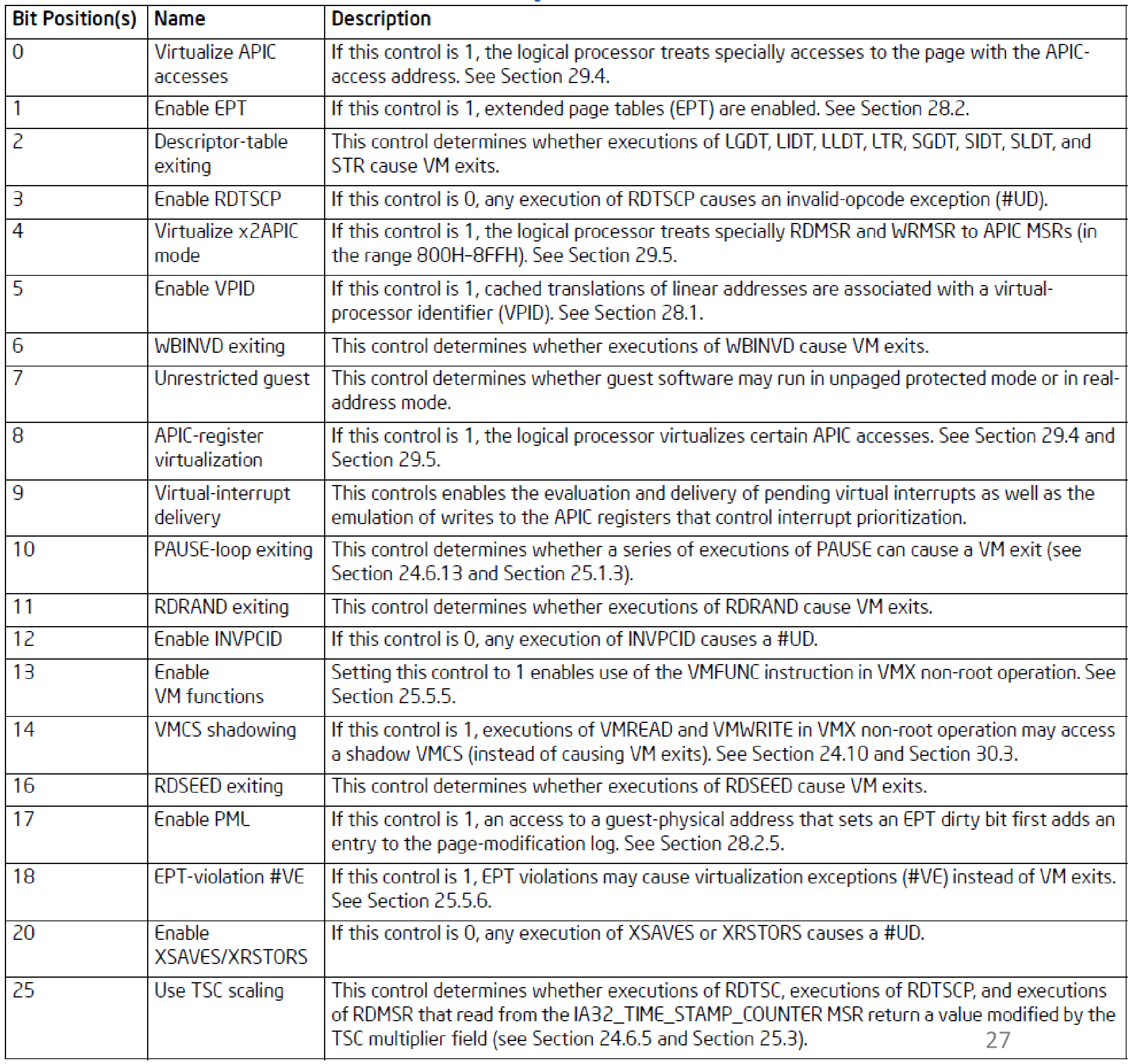


SKL031	A VMX Transition Attempting to Load a Non-Existent MSR May Result in a Shutdown
Problem	A VMX transition may result in a shutdown (without generating a machine-check event) if a non-existent MSR is included in the associated MSR-load area. When such a shutdown occurs, a machine check error will be logged with IA32_MCi_STATUS.MCACOD (bits [15:0]) of 406H, but the processor does not issue the special shutdown cycle. A hardware reset must be used to restart the processor.
Implication	Due to this erratum, the hypervisor may experience an unexpected shutdown.
Workaround	Software should not configure VMX transitions to load non-existent MSRs.
Status	For the steppings affected, see the Summary Table of Changes.

Source: http://www.intel.com/content/www/us/en/processors/core/desktop-6th-gen-core-family-spec-update.html

Attack Surface – Rootkits

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



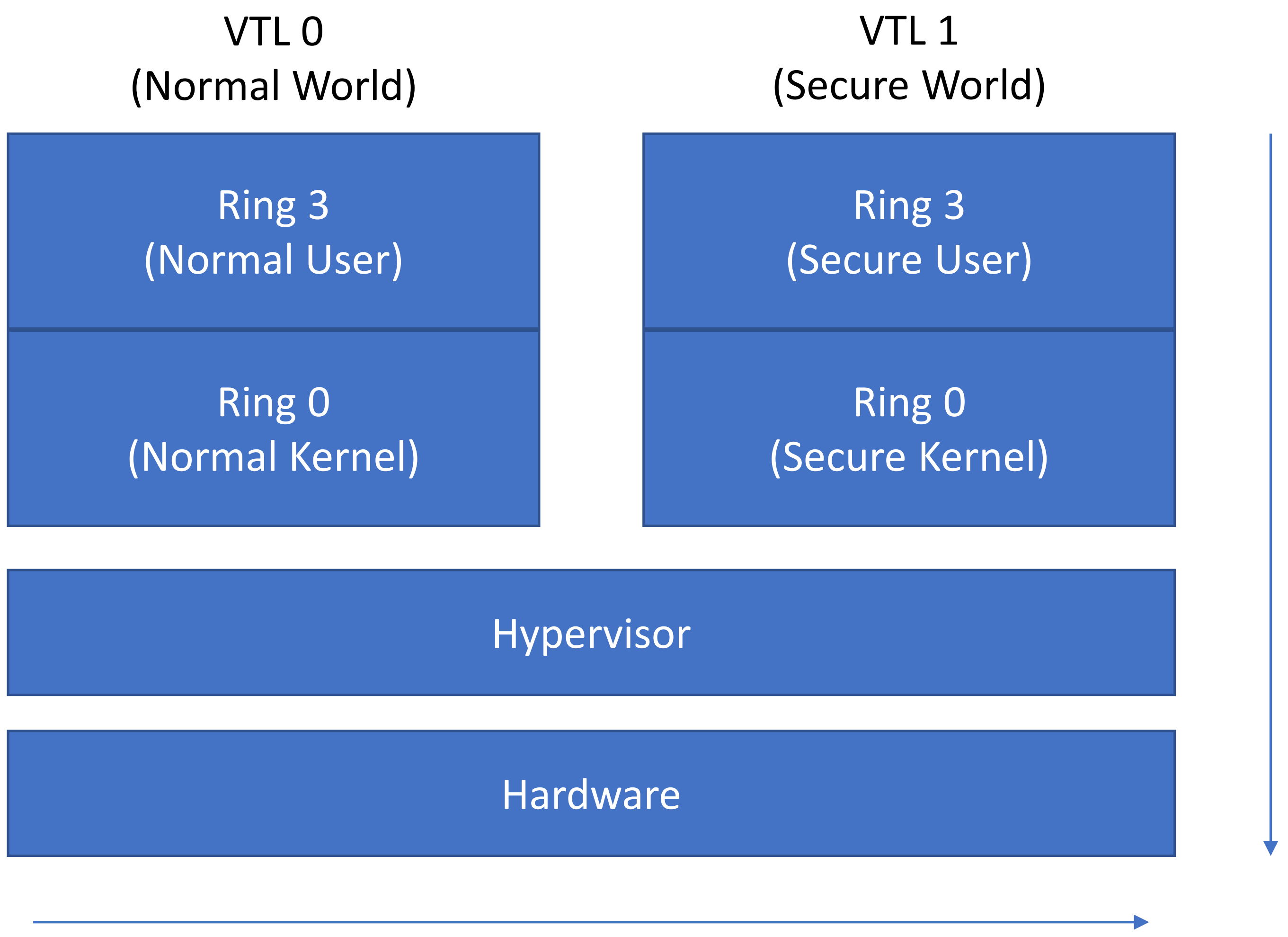
Bit Position(s)	Name	Description
0	Virtualize APIC accesses	If this control is 1, the logical processor treats specially accesses to the page with the APIC-access address. See Section 29.4.
1	Enable EPT	If this control is 1, extended page tables (EPT) are enabled. See Section 28.2.
2	Descriptor-table 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.
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.
20	Enable XSAVES/XRSTORS	If this control is 0, any execution of XSAVES or XRSTORS causes a #UD.
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).

Source: http://www.intel.com/content/www/us/en/processors/architectures-software-developer-manuals.html

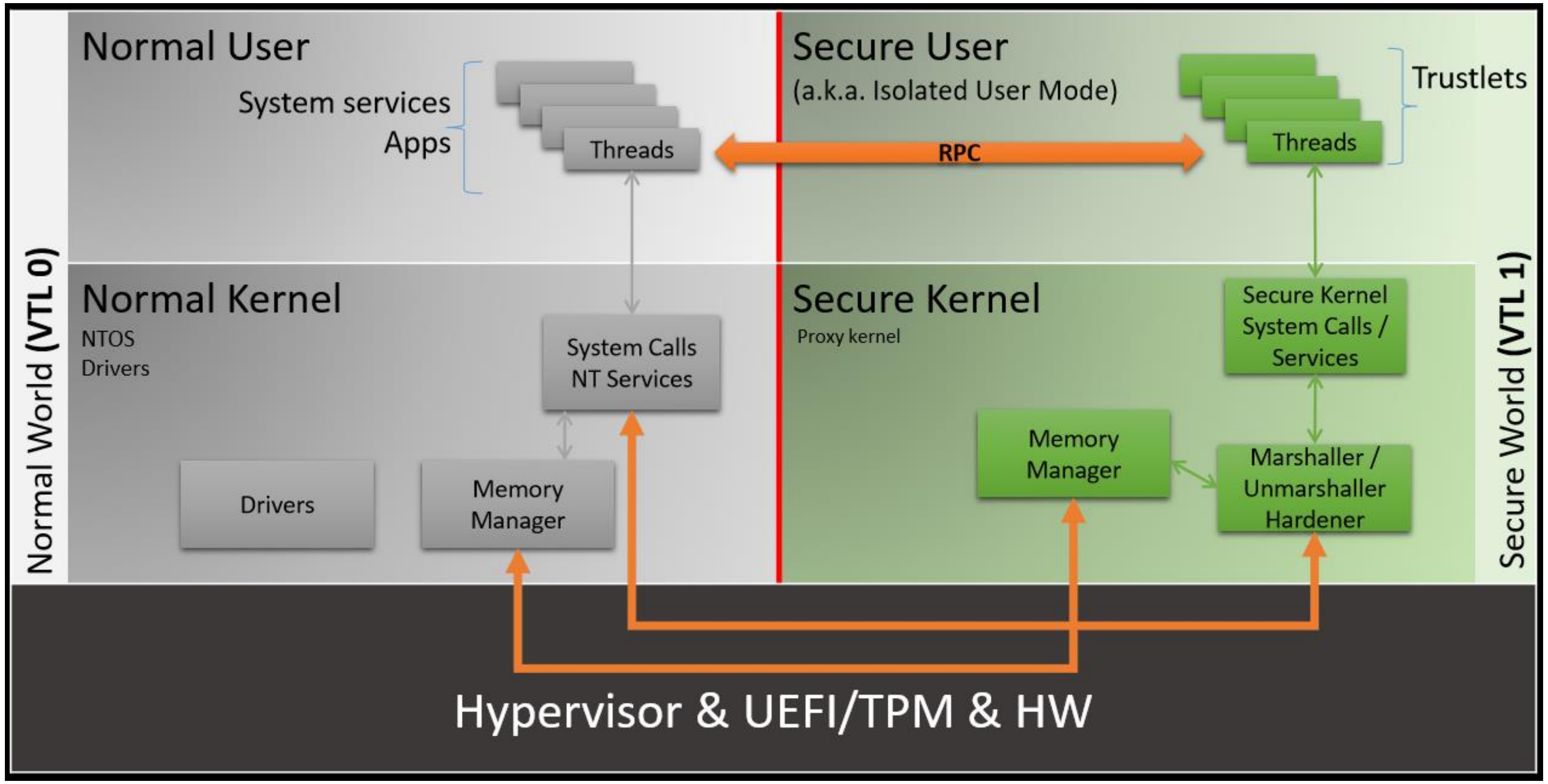
Agenda

- Introduction
- Architectural review
- Attack surface discussion
- Leveraging virtualization for security improvement
- Conclusion

Microsoft Virtualization Based Security (VBS) – Overview



Microsoft Virtualization Based Security (VBS) – Overview

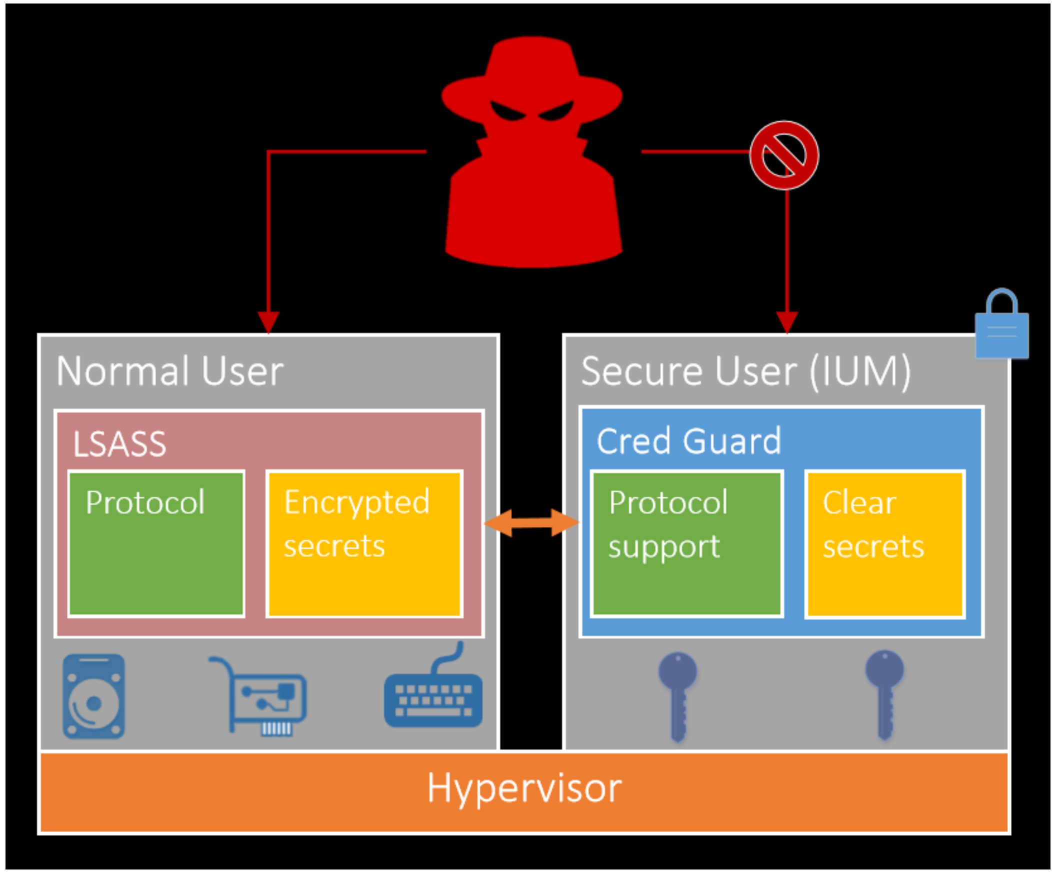


Source: Black Hat 2015 – "Defeating Pass-the-Hash – Separation of Powers" - https://www.blackhat.com/docs/us-15/materials/us-15-Moore-Defeating%20Pass-the-Hash-Separation-Of-Powers.pdf

Microsoft VBS – Attack Surface Brainstorm

- Host -> Hypervisor
- RPC
- System calls
- Memory manager

Microsoft Credential Guard



Source: Black Hat 2015 – "Defeating Pass-the-Hash – Separation of Powers" - https://www.blackhat.com/docs/us-15/materials/us-15-Moore-Defeating%20Pass-the-Hash-Separation-Of-Powers.pdf

Agenda

Conclusion

- As any other piece of software, virtualization can also have vulnerabilities
- Virtualization can be leveraged to improve the security of systems

Thanks!!!

Gabriel Negreira Barbosa
ganegrei [at] microsoft [dot] com