# A Hypervisor IPS based on Hardware Assisted Virtualization Technology

Fourteenforty Research Institute, Inc.
**http://www.fourteenforty.jp**

Senior Research Engineer
Junichi Murakami


# Presentation Outline

1. Review of subversive techniques in kernel space
2. Review of Virtualization Technology
3. Viton, Hypervisor IPS
4. Conclusions

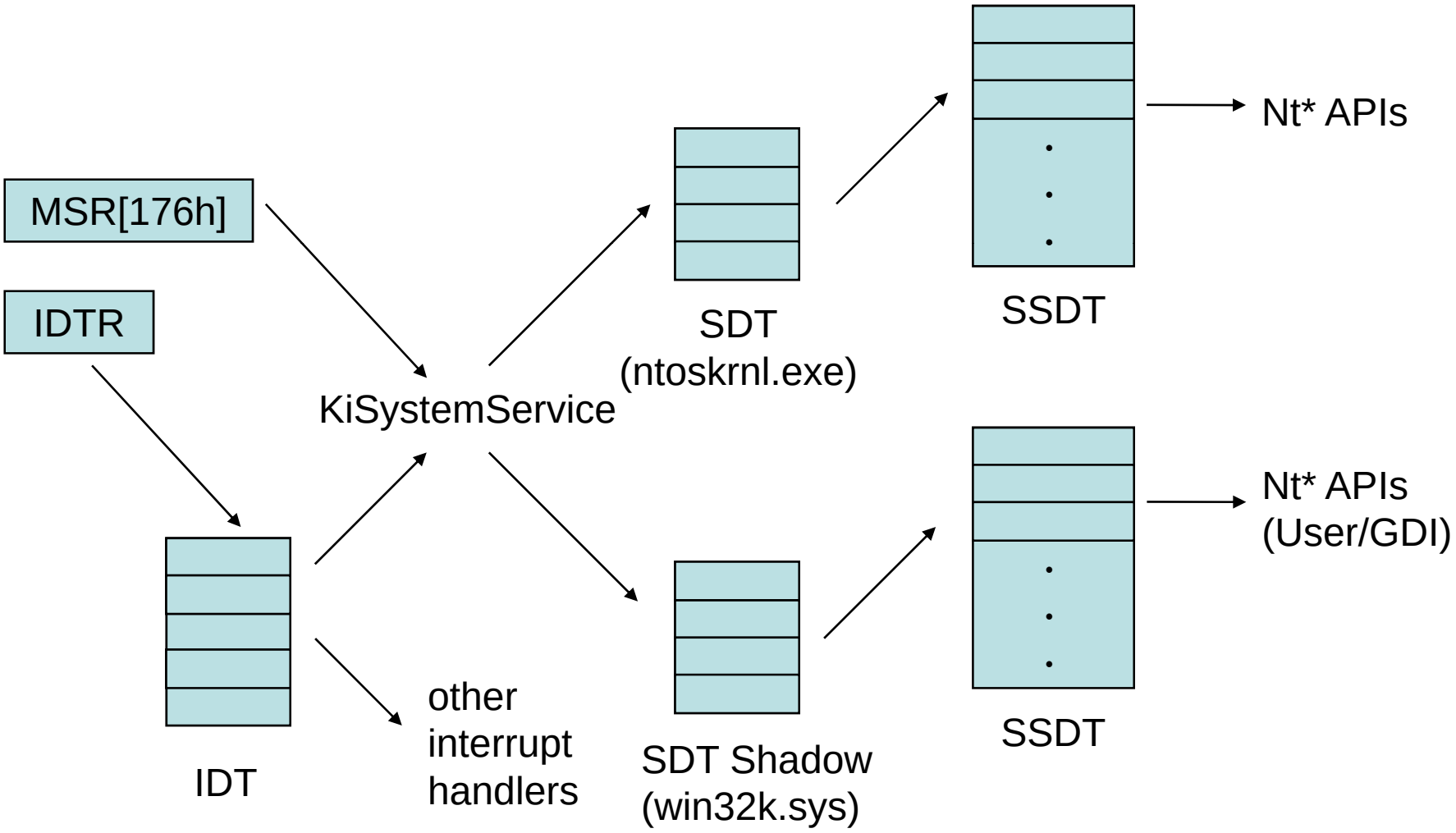

1. Review of subversive techniques in kernel space


# Remember Joanna's classification

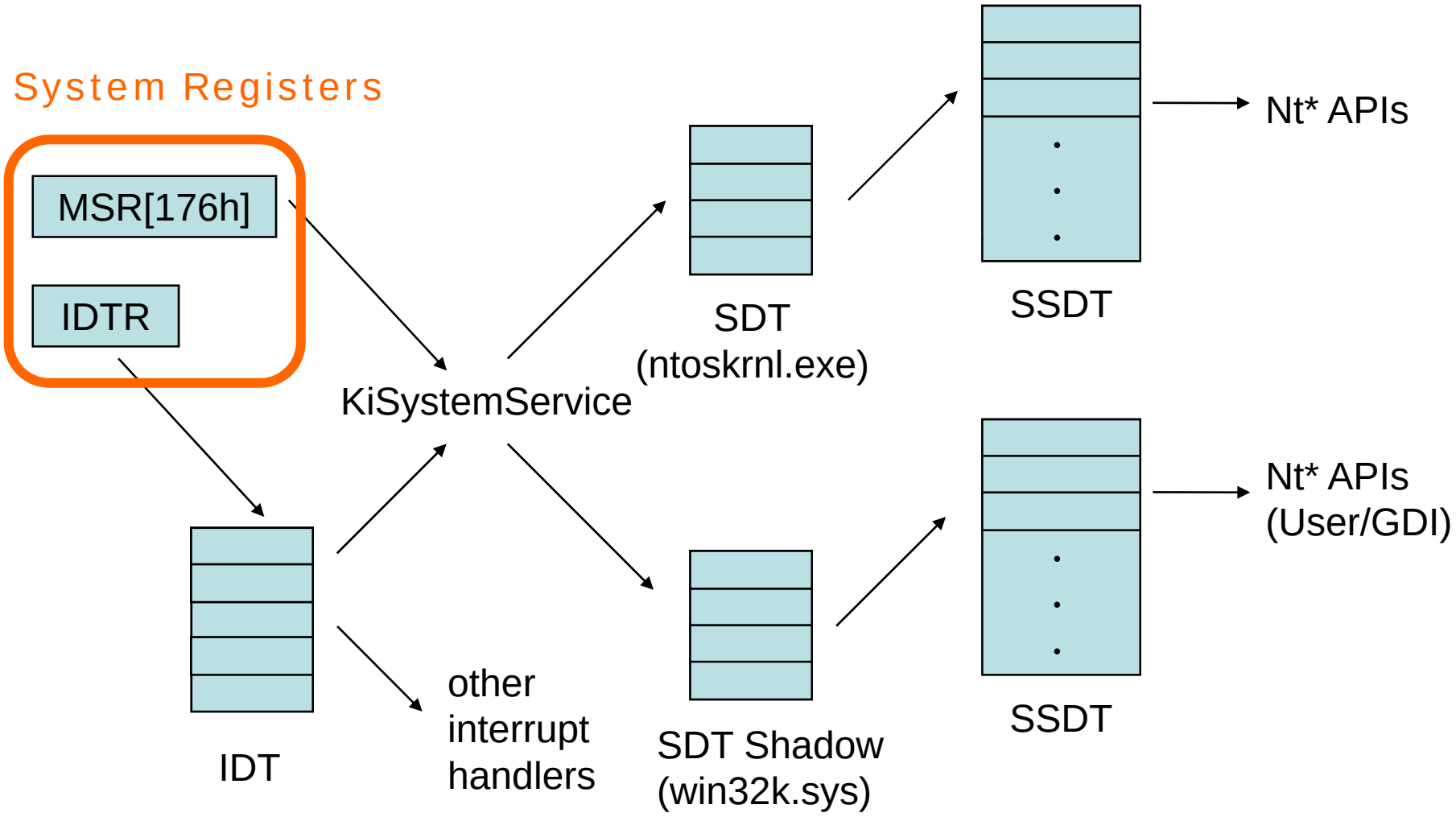
- Joanna Rutkowska proposed stealth malware taxonomy in November, 2006.
  http://invisiblethings.org/papers/malware-taxonomy.pdf

- Type 0
  - standalone malware, which never changes any system resources
- Type I
  - changes the persistent system resources
- Type II
  - changes the non-persistent system resources
- Type III
  - malware runs outside the system

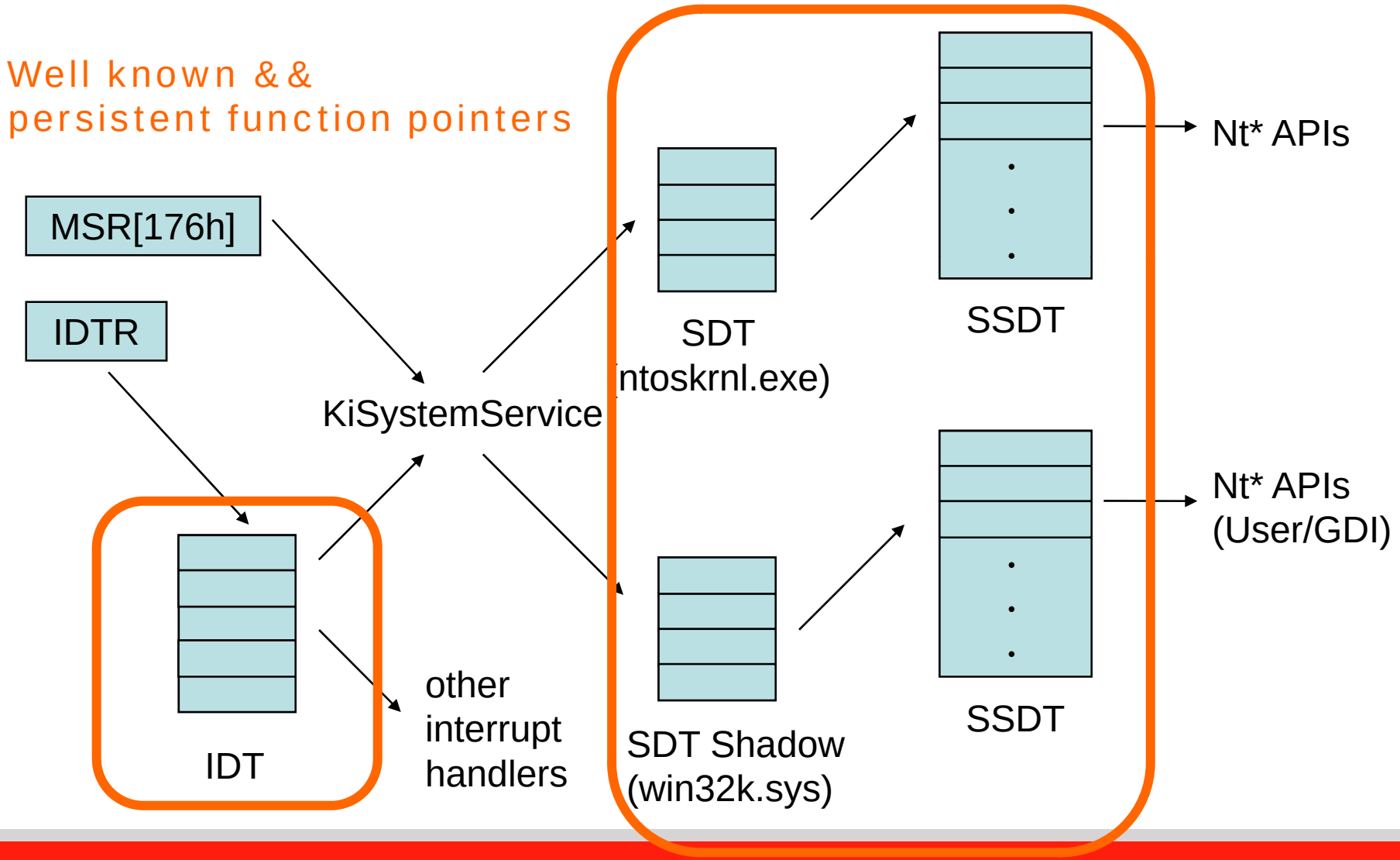
# Type I: Overview of Hooking Points



# Type I: Overview of Hooking Points

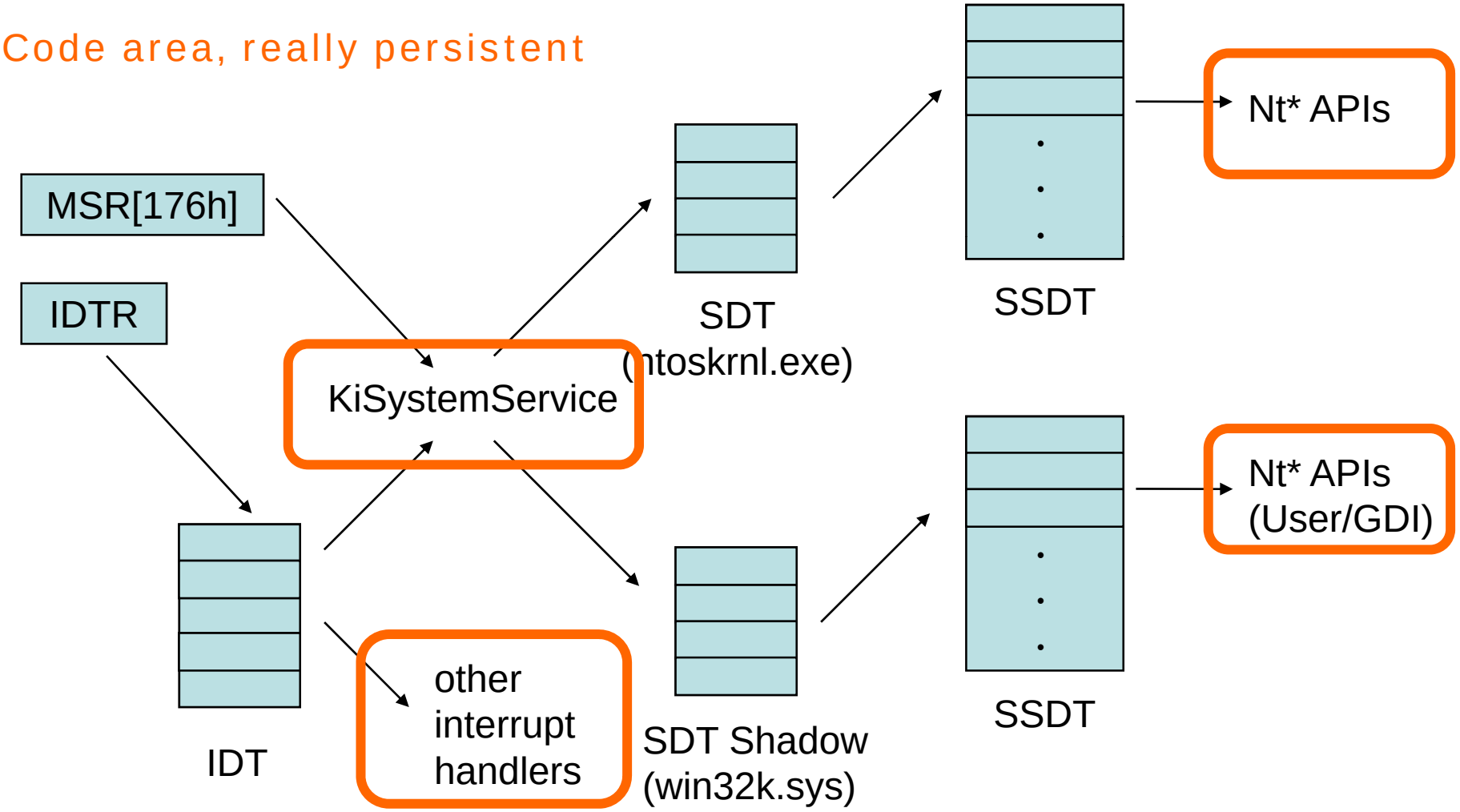


# Type I: Overview of Hooking Points



# Type I: Overview of Hooking Points

Code area, really persistent




# Type I

- It is easy to detect
- PatchGuard in Vista(x64) is a countermeasure for this type
- Many rootkit detectors have been released for this type


# Type II

- Hooker changes the non-persistent system resources
- Hooking point might be modified by the regular execution path

- DKOM(Direct Kernel Object Manipulation)
  - by http://www.blackhat.com/presentations/win-usa-04/bh-win-04-butler.pdf

- KOH(Kernel Object Hooking)
  - by Greg Hoglund in Jan, 2006 http://www.rootkit.com/newsread.php?newsid=501


# DKOM

- Hooker manipulates the process list, tokens and other kernel objects directly

- For example:
  - Unlink target process from process list
  - Add/remove priviledges to tokens

- DKOM's possibilities are limited
  - Whether information hiding can be done depends on the implementation of process that deals with the data

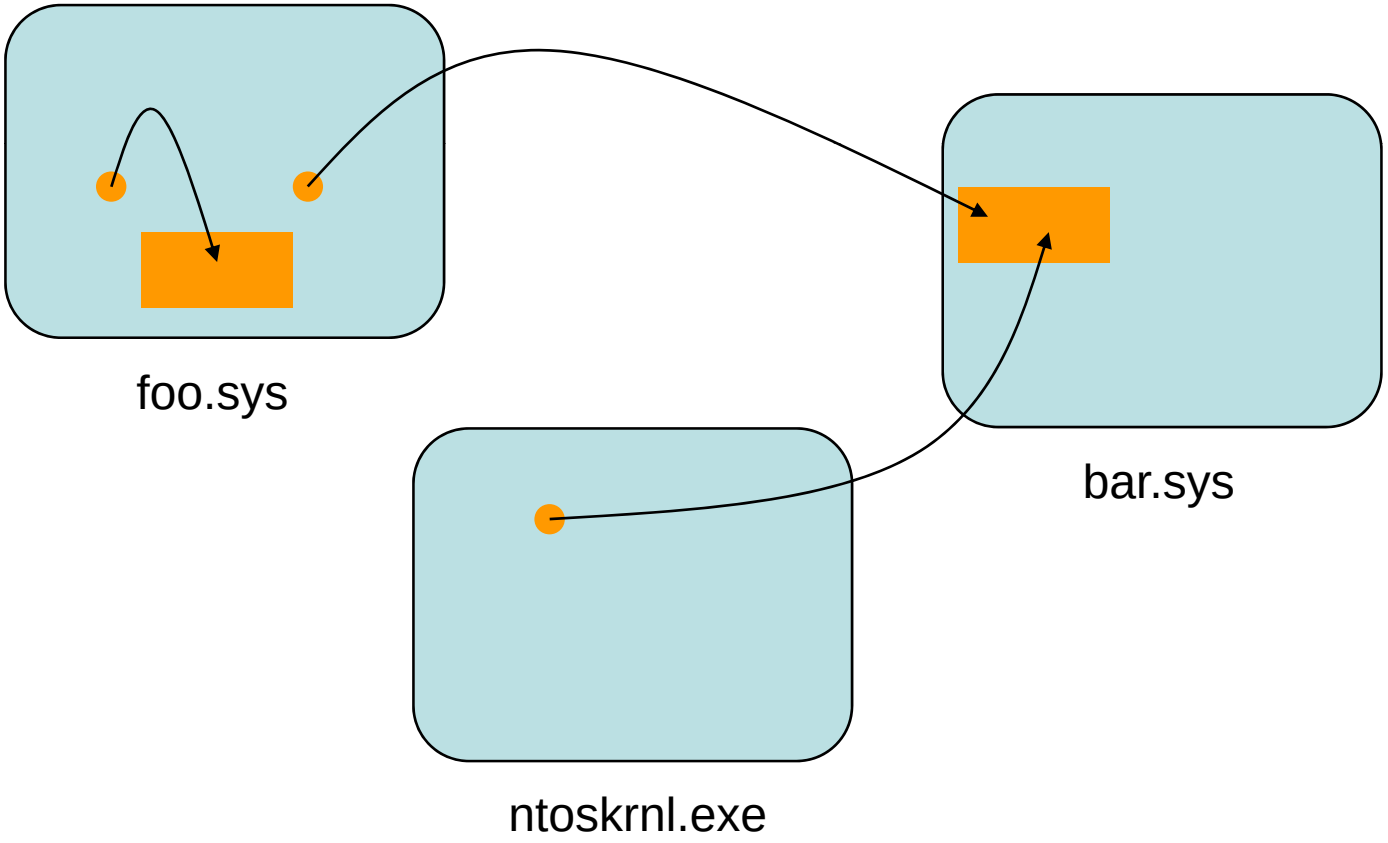

# KOH

- Remember the SDT, SSDT and other well known && persistent function pointers?

- Do you known how many such patching points are there in kernel space?
  - They might or might not be persistent
  - It depends on each kernel object

- Detector has to understand all function pointers
- is_within_own_memory_range(PVOID Address) is useful, but not enough

# is_within_own_memory_range(PVOID Addr)




# Type III

- No hooker exists in the system(guest)
- Malware (ab)uses Virtualization Technology
- SMM Rootkit and Firmware Rootkit might also fall into this category (a problem of taxonomy that is not important for our cause)

- BluePill
  - Original BP was presented by Joanna Rutkowska in BH-US-2006.
  - (Current) New BP supports both Intel VT and AMD-v technologies,
    and is also capable of on the fly loading and unloading
  - BP doesn't modify any system resources on the guest
    - From a technical view, BP patches the guest's PTE to hide its loaded virtual memory from the guest
    - However this doesn't really help detecting it


# Type III (cont.)

- Vitriol
  - Presented by Dino Dai Zovi, Black Hat US 2006
  - VT-x rootkit, closed source

- VMM Rootkit Framework
  - Posted by Shawn Embleton, Aug, 2007
    http://www.rootkit.com/newsread.php?newsid=758
  - This is really good start point for learning for how to create VMM


# Case Study: Storm Worm

- The Storm Worm first appeared in Fall, 2006
- Some variants have rootkit functions to hide from AV products
- As of Jan 2008 we can see "Happy New Year 2008" variants
- When a user clicks onto the executable,


# Storm Worm

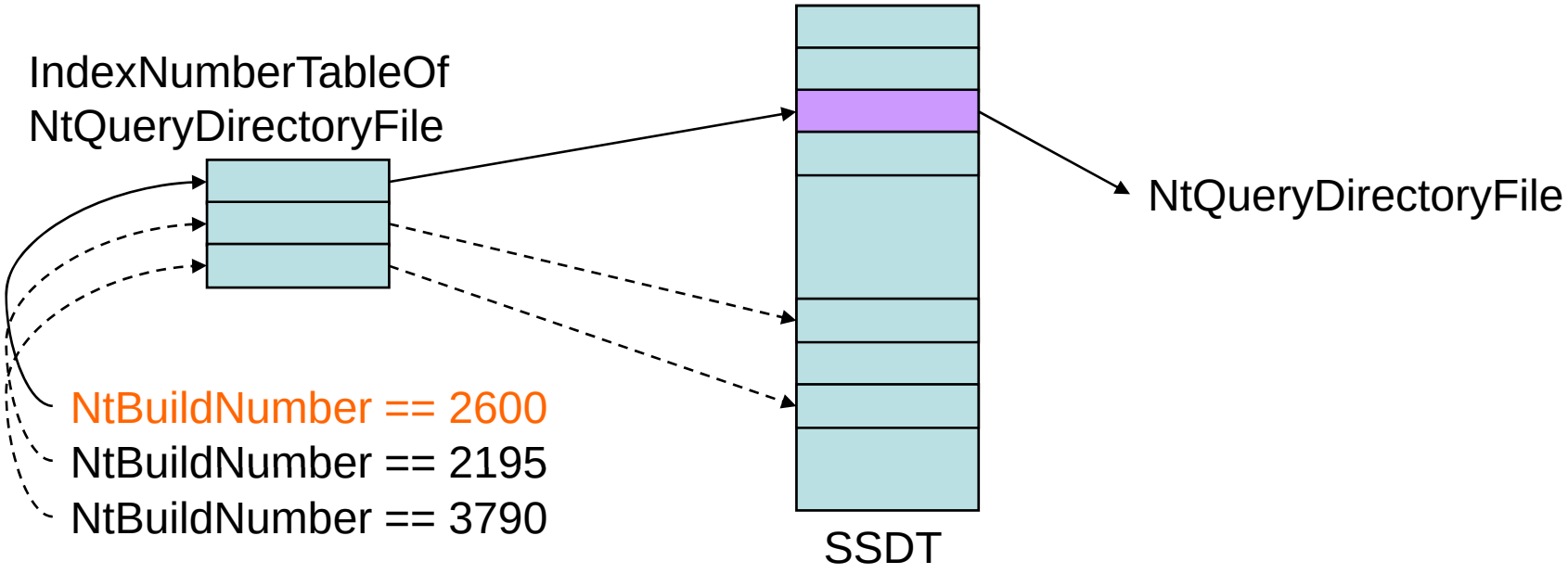
1. Executable drops the system driver (.sys), and loads it into the kernel using Service Control Manager (SCM)

2. Driver has two functions shown below
   - Rootkit functions
     Hide files, registry entries and connections using SSDT and IRP hooking
   - Code Injection function
     Inject malicious code (not DLL) into process context of services.exe and execute it

3. Injected code starts P2P communication

# Rootkit functions

- Storm Worm hooks three Native APIs
  - NtQueryDirectoryFile, NtEnumerateKey, NtEnumerateValueKey
- API Index of SSDT is different for each NtBuildNumber
- Storm Worm has index number tables for build 2195(2k), 2600(XP) and 3790(2k3)




# Rootkit functions (cont.)

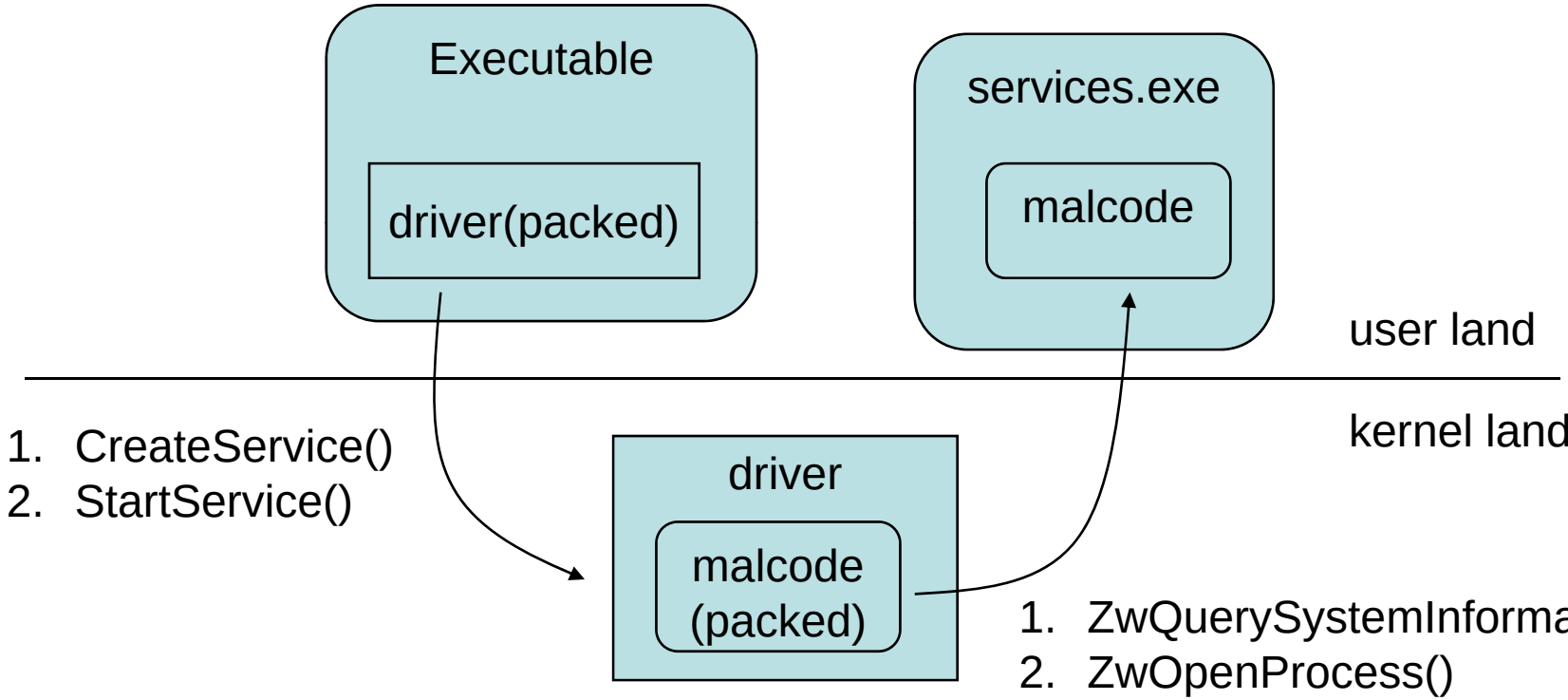
- It hooks the IRP_DEVICE_CONTROL routine by patching the TCP DriverObject's IRP table ("\\Device\\Tcp")
- Hide connections from netstat

But is this KOH?

YES: It modifies the IRP Table contained within the DriverObject

NO: Many people know about the existence of  IRP tables

# Code injection function




1. CreateService()
2. StartService()

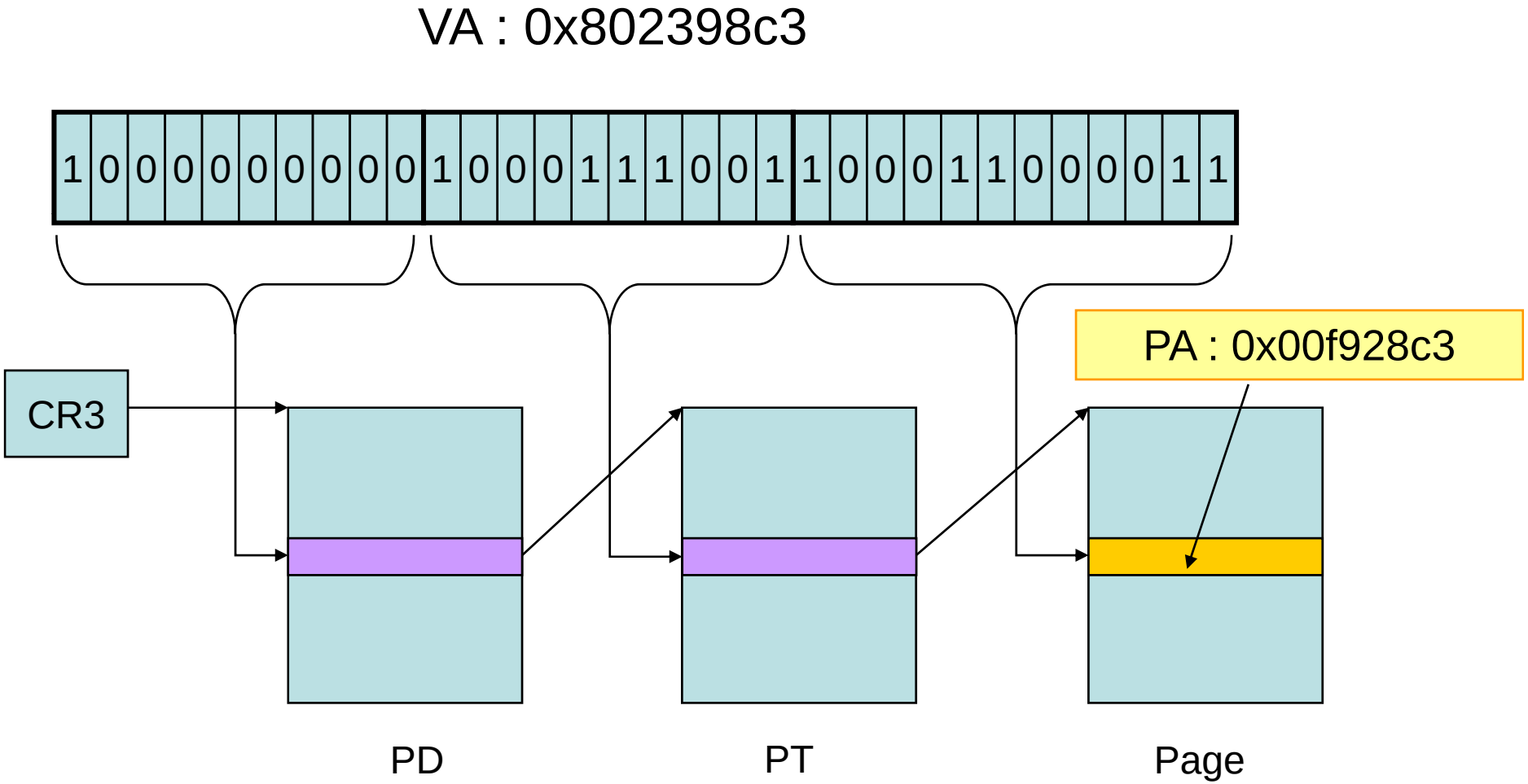
1. ZwQuerySystemInformation()
2. ZwOpenProcess()
3. ZwAllocateVirtualMemory()
4. (patch to thread object)
5. KeInitializeApc()
6. KeInsertQueueApc()


# 2. Review of Virtualization Technology

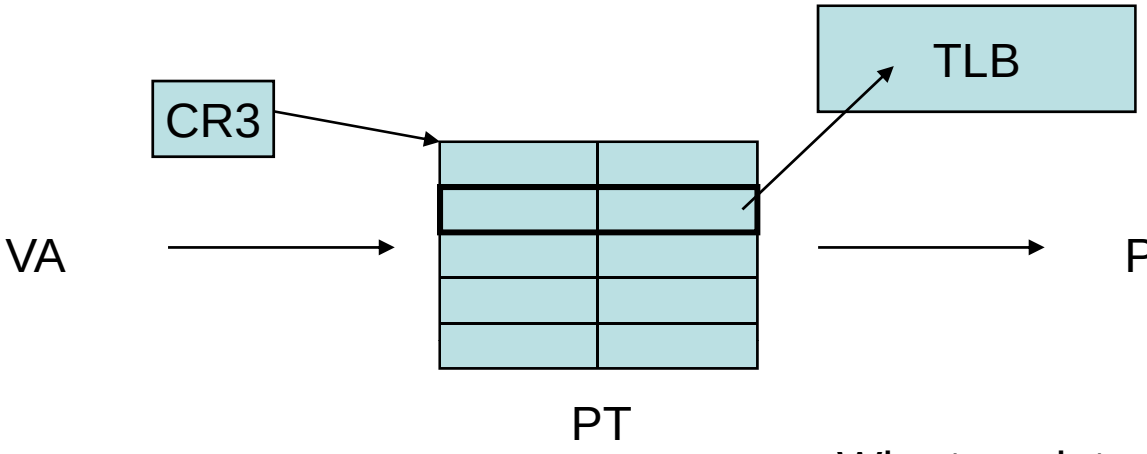
# What we have to consider "Virtualization"

- CPU Virtualization
  - Some registers should be reserved for VMM and each VM. GDTR, LDTR, IDTR, CR0-4, DR0-7, MSR, Segment Register, etc
  - Exceptions
- Memory Virtualization
  - should separate VMM memory space and each VM's memory space
- Device Virtualization
  - Interrupt, I/O instructions, MMIO, DMA access

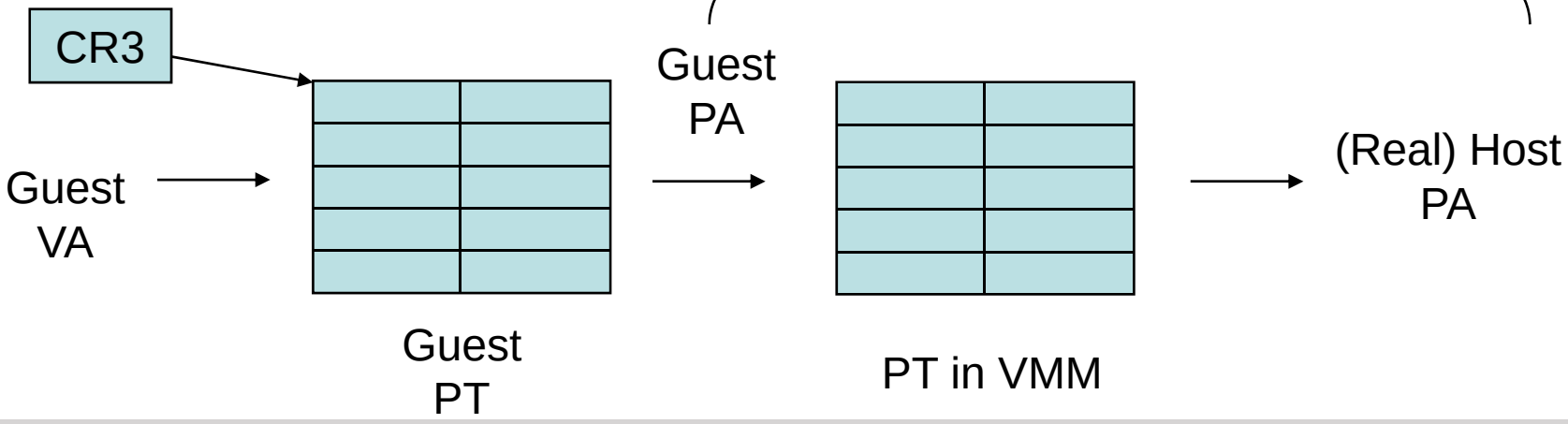
# VA -> PA


VA : 0x802398c3
1 0 0 0 0 0 0 0 0 0 1 0 0 0 1 1 1 0 0 1 1 0 0 0 1 1 0 0 0 0 1 1
PA : 0x00f928c3
CR3
PD
PT
Page

# To simplify...




Who translates this?

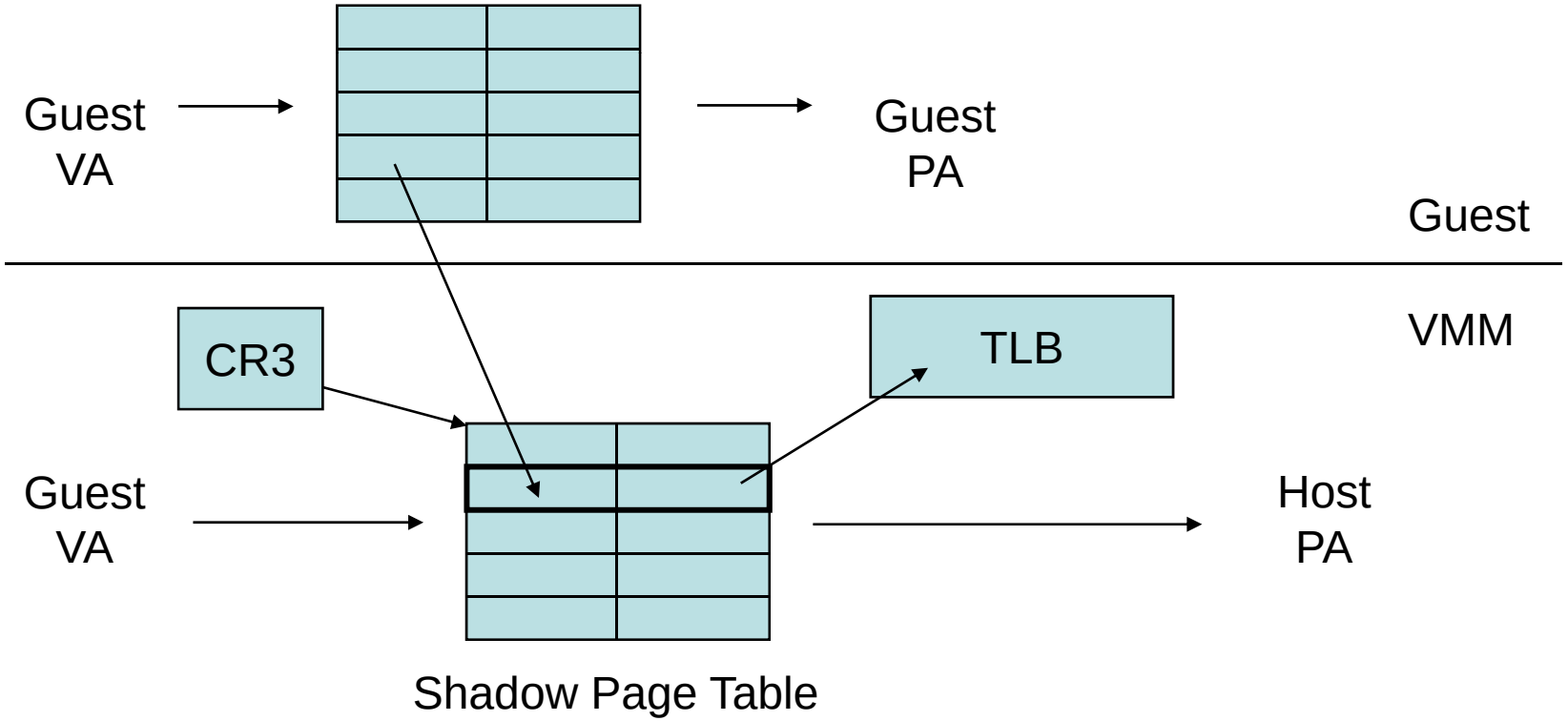



# Memory virtualization

- MMU does the address translation according to CR3
- If the processor supports EPT (Extended Page Table), this 2-stages translation is automatically done by the MMU
  - EPT is not implemented yet
- VMM should implement this translation as software using Shadow Paging

# Shadow Paging

- VMM updates SPT on #PF in the guest
  - and also emulates TLB flush caused by MOV to CR3 and INVLPG

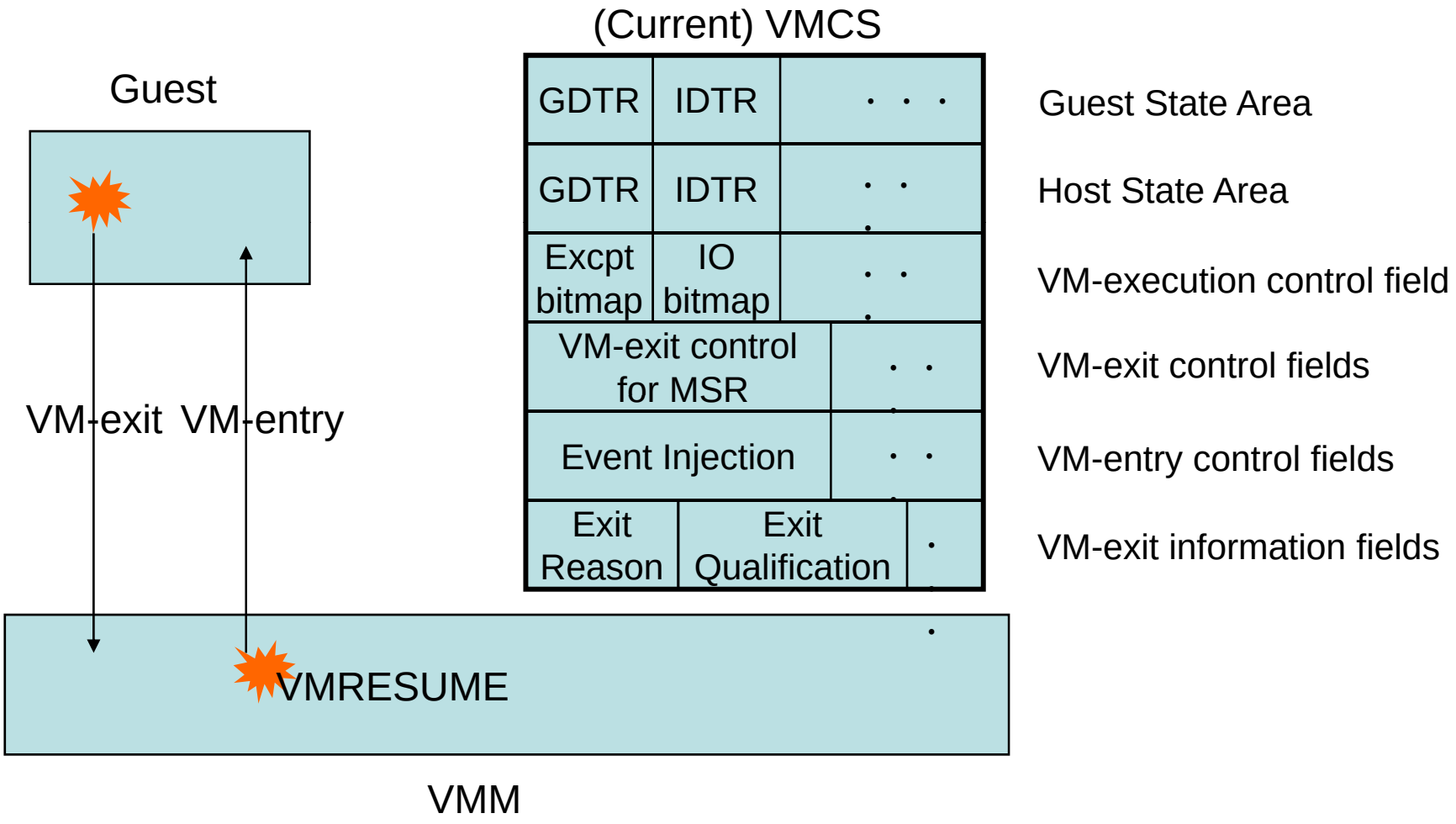


# Intel VT

- Intel VT is the Intel VT-* family's generic name
  - VT-x, virtualization for x86/64
  - VT-d, virtualization for device (Directed I/O)
  - VT-i, virtualization for Itanium

- Key factors
  - VMX mode
    - VMX root-operations(ring0-3)
    - VMX non-root-operations(ring0-3)
  - VMCS (Virtual Machine Control Structure)
  - VMX Instructions set
    - VMXON, VMXOFF, VMLAUNCH, VMRESUME, VMCALL, VMWRITE, VMREAD, VMCLEAR, VMPTRLD, VMPTRST

# How Intel VT works:



# enum EXIT_REASON {

- Specific instructions
  - CPUID, INVD, INVLPG, RDTSC, RDPMC, HLT, etc.
- I/O Instructions
  - IN, OUT, etc.
- All VMX Instructions
- Exceptions/Interrupts
- Access to CR0-CR4, DR0-DR7, MSR
- SMI/RSM
- etc.

};

# Steps to launch the VM and VMM

- Confirm that the processor supports VMX operations
  - CPUID
- Confirm that VMX operations are not disabled in the BIOS
  - MSR_IA32_FEATURE_CONTROL
- Set the CR4.VMXE bit
- Allocate and Initialize VMXON region
  - Write lower 32 bits value of VMX_BASIC_MSR to VMXON region
- Execute VMXON
  - CR0.PE, CR0.PG, and CR4.VME must be set.


# Steps to launch the VM and VMM (cont.)

- Allocate VMCS regions
- Execute VMPTRLD to set Current VMCS
- Initialize Current VMCS using VMREAD and VMWRITE
  - VMCS contains the EP of VMM, and Guest IP after VMLAUNCH
- Execute VMLAUNCH
  - Continue to execute the guest from IP is contained in VMCS
- When VM-exit occurred, IP and other registers are switched to VMM ones.


# 3. Viton, Hypervisor IPS


# Viton

- IPS, which runs outside the guest
- Just a PoC, only tested on Windows XP SP2
- Forces immutability to persistent system resources
- Observes control and system registers modification - VMX instructions are raised in the guest

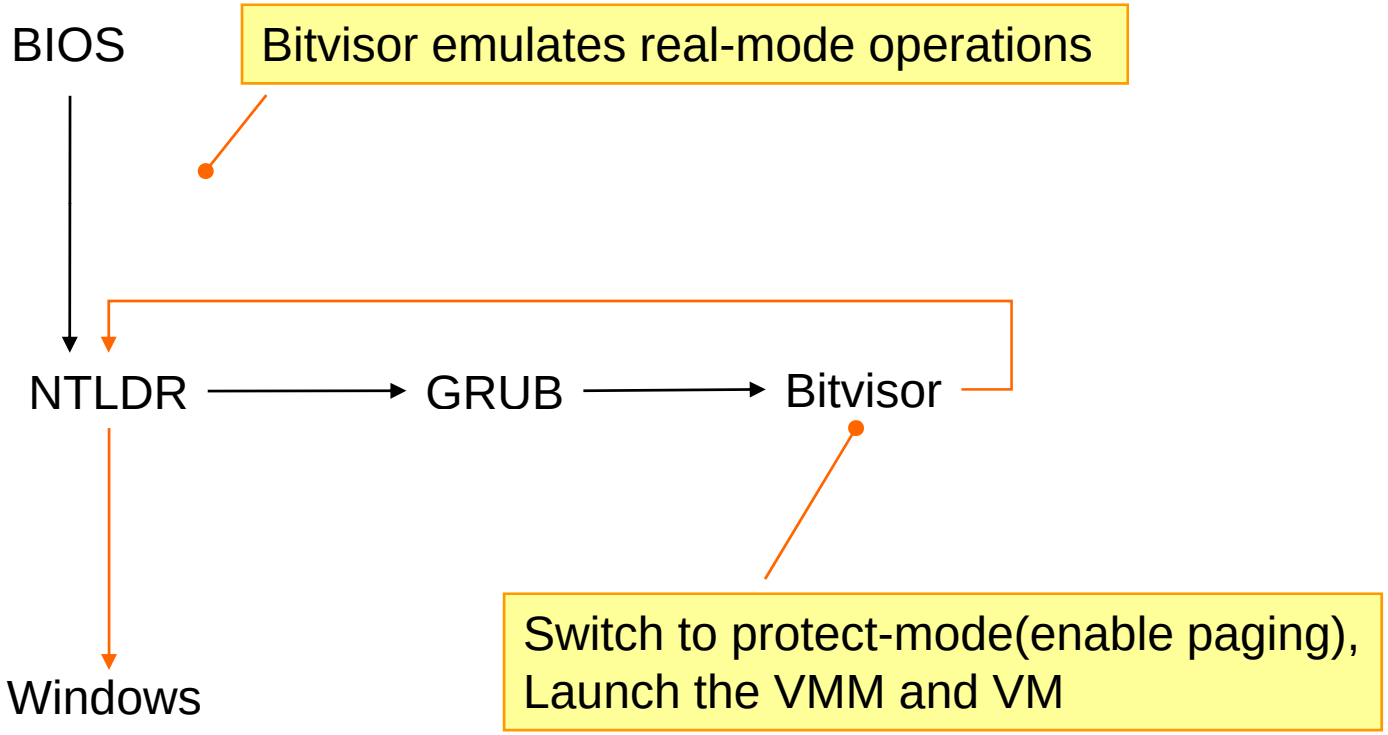
- It is based on **Bitvisor**


# Bitvisor - http://www.securevm.org

- The Bitvisor VMM software is developed by the Secure VM project centered around Tsukuba Univ. in Japan

- Features:
  - Open source, BSD License
  - Semi-path through model
  - Type I VMM (Hypervisor model, like Xen)
  - Full scratched, pure domestic production
  - Support for 32/64 bits architecture in VMM
  - Support for Multi-core/processor in VMM and Guest
  - Can run Windows XP/Vista as Guests without modification
  - Support for PAE in the Guest
  - Support for Real-mode emulation

# How Bitvisor works: Launch process

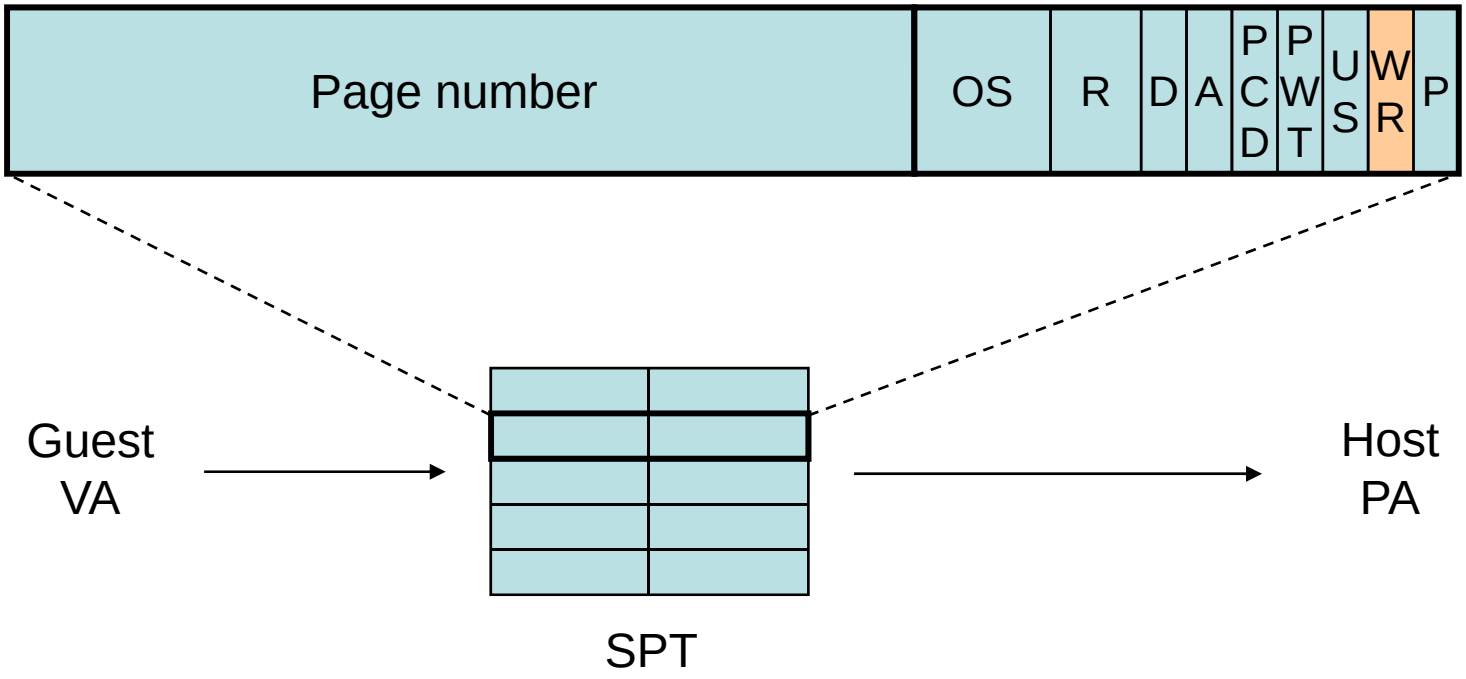



# What Viton protects/detects:

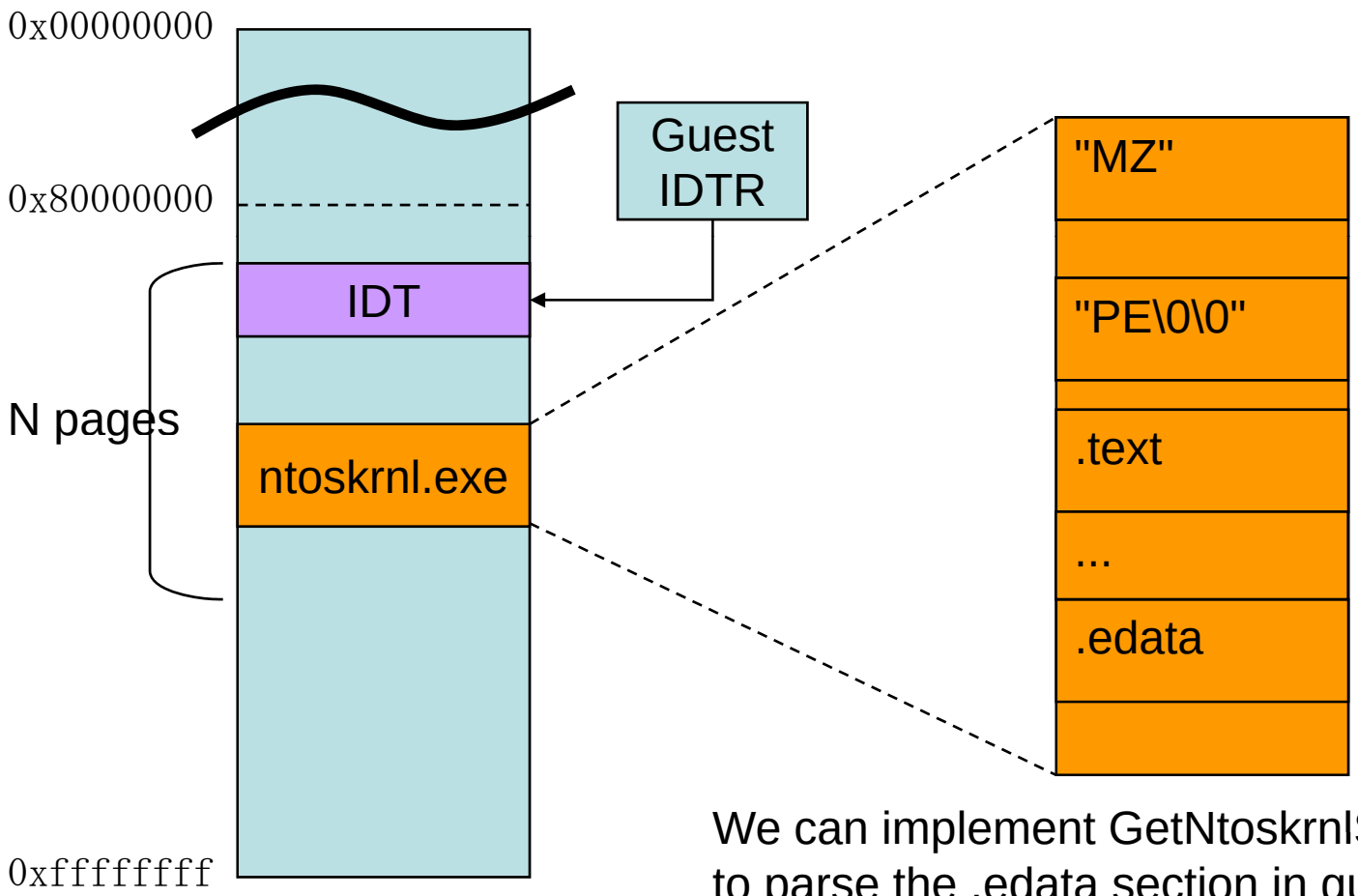
- Instructions
  - Detect and block all VMX Instructions
- Registers
  - Watchdog for IDTR
  - Locking the MSR[SYSTENR_EIP]
  - Locking the CR0.WP Bit
- Memory
  - Protect from modification
    - All code sections (R-X) in ntoskrnl.exe
    - IDT
    - SDT
    - SDT.ST (SSDT)

# How to protect the guest memory modification

- Viton clears the WR bit in a SPT entry
  - If CR0.WP is set, even the kernel cannot modify the page



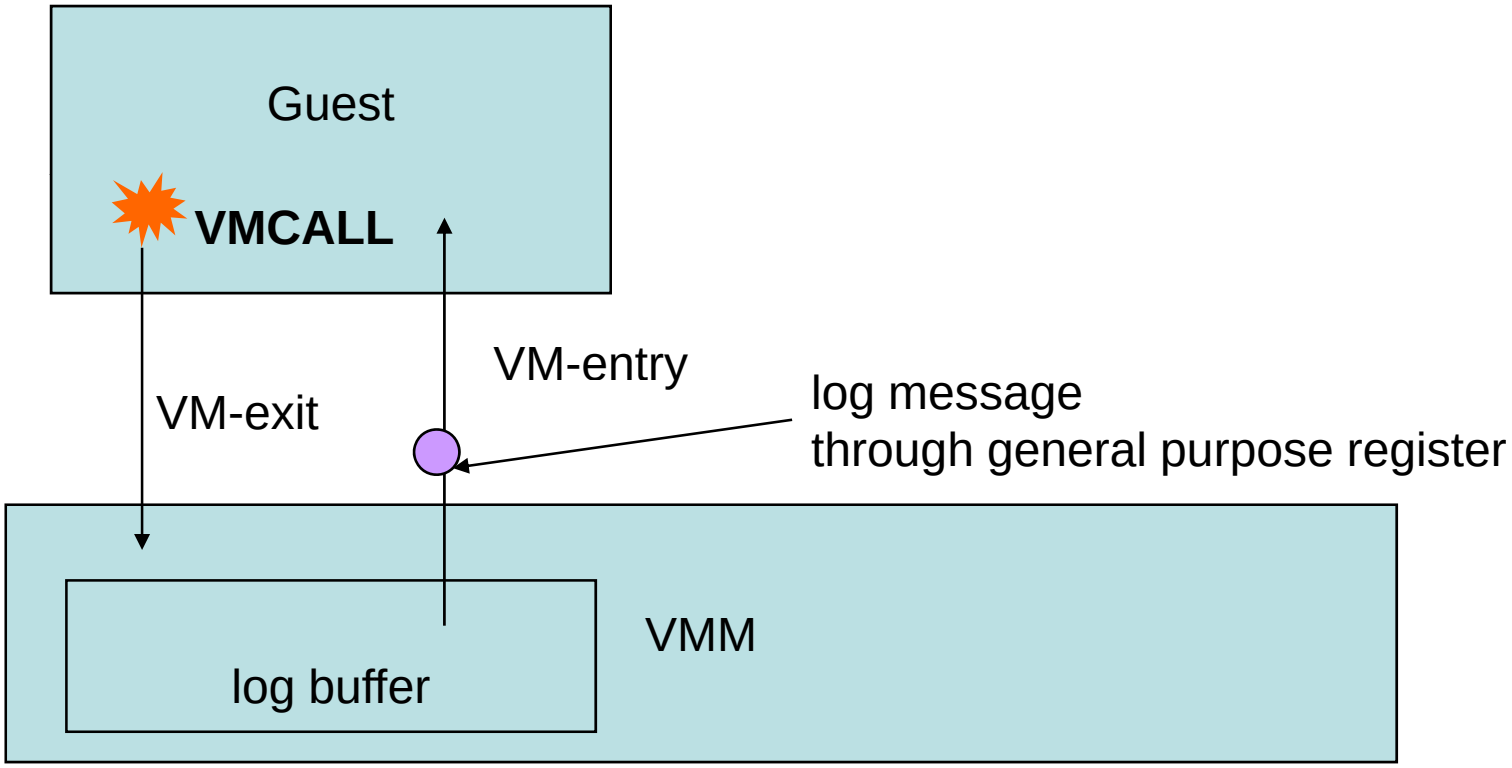
# How to recognize the guest memory layout




We can implement GetNtoskrnlSymbolAddr()
to parse the .edata section in guest VA space.


# Demo

# dbgsh (Bitvisor's debuging function)



# Viton vs.

- Type I
  - Easy
- Type II
  - Difficult
- Type III
  - Easy


# 4. Conclusions

- Virtualization Technology becomes a help to protect the kernel.
  - We can block memory modification.

- However, it is not a silver bullet.
  - Foundation for existing security solutions

# Thank you!


Fourteenforty Research Institute, Inc.
**http://www.fourteenforty.jp**

Senior Research Engineer
Junichi Murakami <murakami@fourteenforty.jp>