How to Implement a Virtual Machine Based Rootkit

BING SUN LinkTrust

taoshaixiaoyao@hotmail.com

Popular “Virtual Execution” Techniques

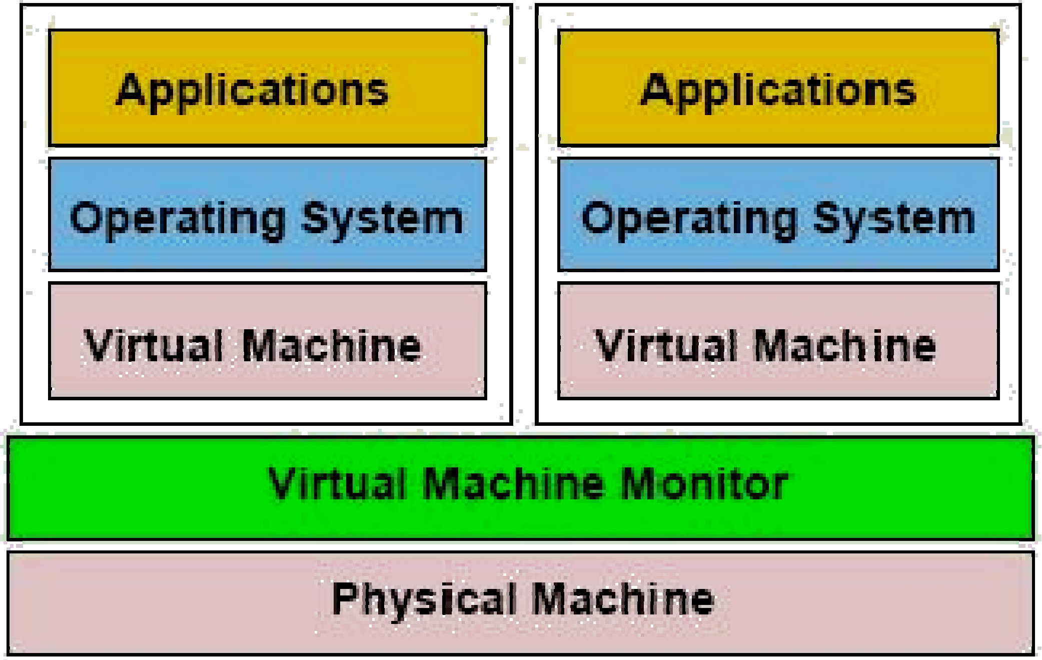
- Pure Emulator: Bochs
- OS/API Emulator: Wine
- Virtual Machine: VMware, Plex86

Full Virtualization vs. Para-Virtualization

- Full Virtualization： Virtualize all features of processor and hardware. Typical representatives include IBM VM/370, VMware.
- Para-Virtualization： May make reasonable assumptions and do some modifications on the Guest OS. Typical representatives include Xen, Denali.

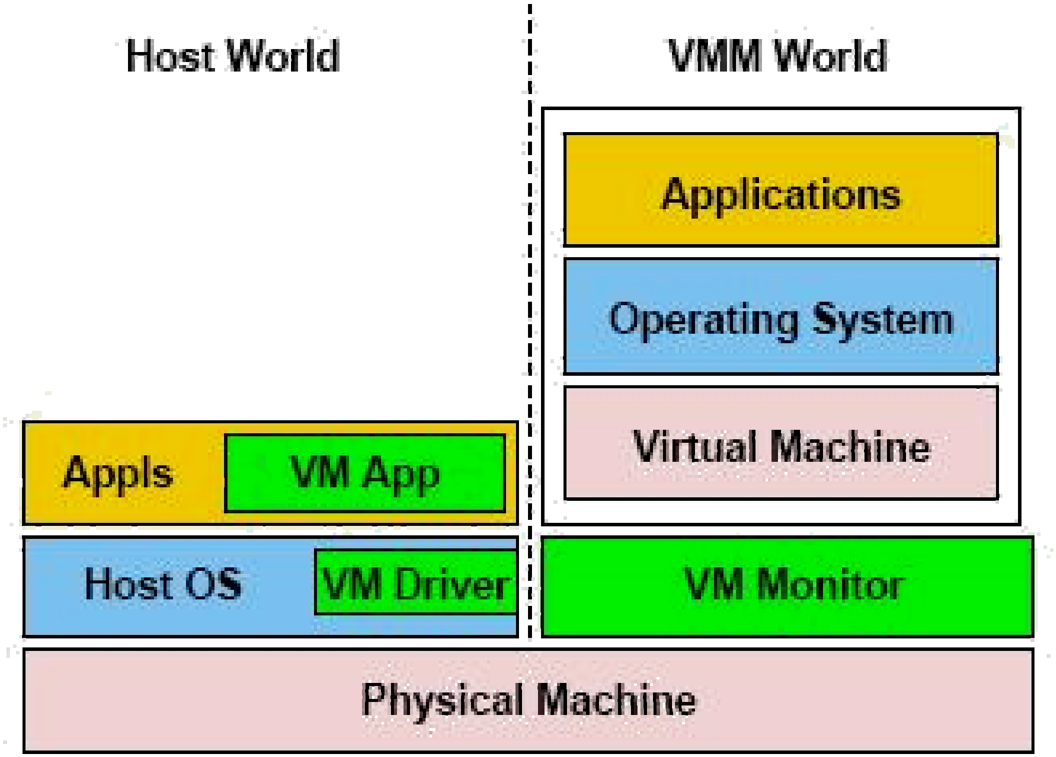
The Classification Of Virtual Machine Monitor

- Type I VMM



The Classification Of Virtual Machine Monitor

- Type II VMM



Hardware Virtualization Support

- Intel VT-x: <<Understanding Intel® Virtualization Technology (VT) >>
- AMD Pacifica : <<Virtualization Technology For AMD Architecture>>

The Standards of Virtualizable Processor Architecture

R. Goldberg’s requirements：

1. At least two processor modes.
2. A method used by non-privileged program to invoke privileged system routines.
3. Memory relocation or protection mechanism.
4. Asynchronous interrupt mechanism.

The Standards of Virtualizable Processor Architecture

John Scott Robin’s standards：

1. The execution manner of non-privileged instructions are almost identical in two modes, the user mode and the privileged mode.
2. Protection mechanism and address translation system to isolate and protect the real machine from the virtual one.
3. Notification on execution of sensitive instructions and the ability to emulate them.

The Challenges On x86 Virtualization

Limitations of hardware & processor

Hardware: designed to be controlled only by one device driver.

x86：its system features are designed to be configured and used by only one OS. In addition, there are：

- Tight-coupled between some non-strictly related mechanisms.
- Hidden part of segment registers.
- Non-trapped sensitive instructions.

The Challenges On x86 Virtualization

Non-trapped sensitive x86 instructions list:

Most of them are segment/eflags bits manipulation instructions

- lar/lsl/verr/verw
- sgdt/sidt/sldt/str
- smsw
- popf/popfd
- pushf/pushfd
- mov r/m, Sreg
- mov Sreg, r/m
- push Sreg
- pop Sreg

The Challenges On x86 Virtualization

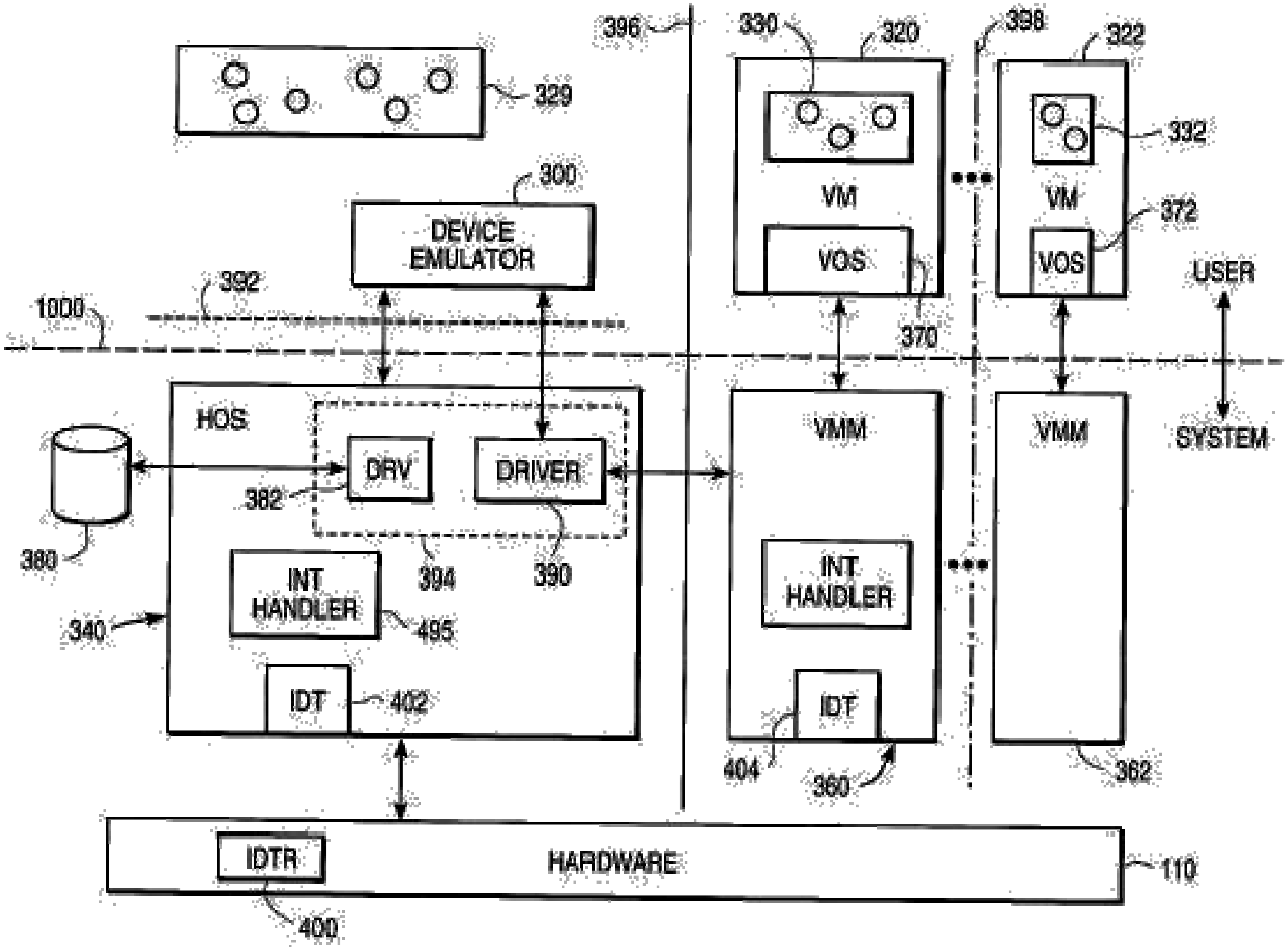
In addition, I/O and control transfer instructions are also related to this topic.

- in/ins/out/outs
- sysenter/sysexit
- call/jmp/int n/ret/iret

The Related Concepts & Terms

- Host OS
- Guest OS
- VM
- VMM
- Hypervisor
- Sensitive Instructions
- Non-Virtualizable Instructions
- Segment Reversibility

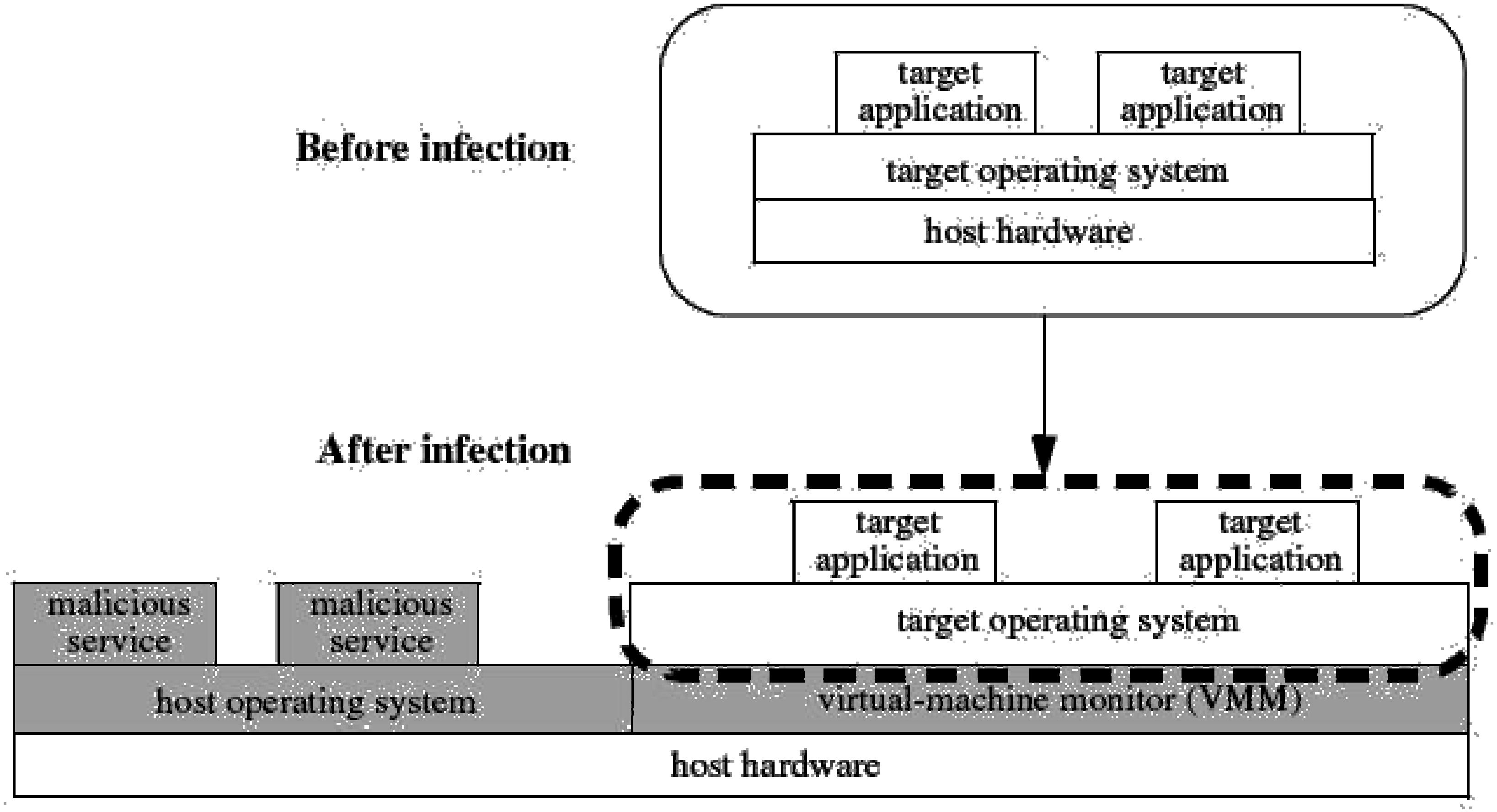
The Whole Structure Of VMware Workstation



VMware’s Working Principle

- Total Context Switch
- x86 Processor Virtualization
- Hardware Devices Virtualization

A VM-Based Rootkit (VMBR) SubVirt

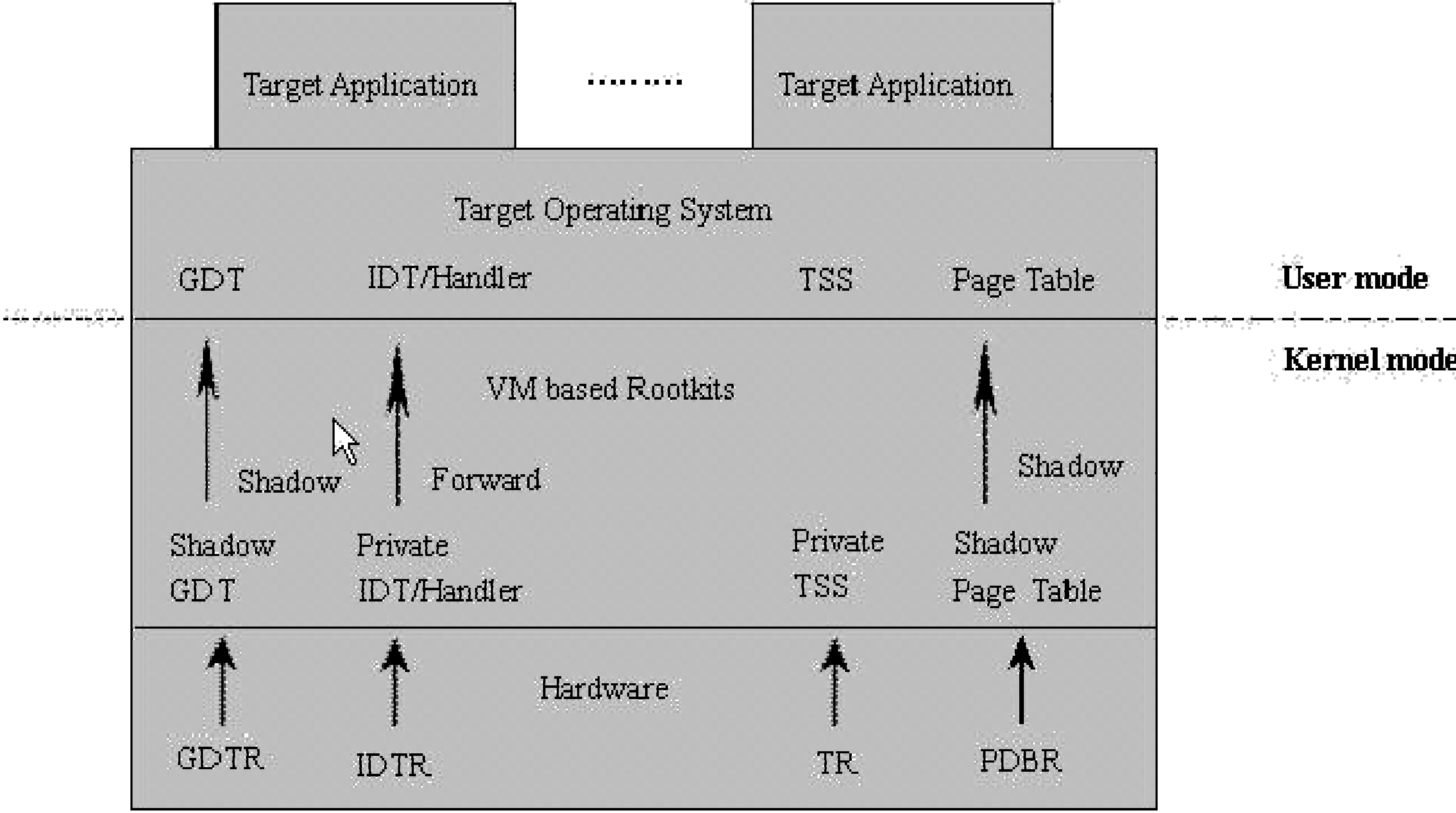


A VM-Based Rootkit (VMBR) SubVirt

- SubVirt, a VM-Based Rookit developed by Microsoft research team and Michigan University.

- Some defects on its practicality :
  - Depends on commercial VM software (VMware or VPC) and a Host OS (Linux).
  - Modifies the system Boot sequence.
  - Emulates a set of different virtual devices.

The Whole Structure Of The New VMBR



The Basic Working Principle Of The New VMBR

- No Host OS and Guest OS, the OS infected is referred to as Target OS.
- Rootkit could be loaded via a kernel mode driver.
- Rootkit will occupy the topmost 4M region in linear address space, and replace the original states in processor with its own ones.
- After being loaded, Rootkit acts as a transparent intermediate lay between the Target OS and real hardware, which is actually a VMM.

x86 4 Modes Virtualization

- Only deal with protected mode presently, the Target’s switching to V86 or SMM mode should be considered in the future for practicability.

- No Binary Translation or Dynamic Scan, use completely Direct Execution for now.

x86 Instruction Execution Virtualization

- Complete Direct Execution plus Privilege Level Compression:
  - Scheme 1: Compress the Target OS kernel mode (ring0) to ring3
  - Scheme 2: Compress the Target OS kernel mode (ring0) to ring1 (recommended)

CPU State Information Virtualization

- Rootkit reserves a small region in its memory space to maintain the Target OS's processor state information, which include general register set, flag register, segment register set, control registers, debug registers, GDTR/LDTR/IDTR/TR, some MSRs, PIC/APIC.
- The opportunity to save sensitive and non-sensitive states.
- Rootkit will use these virtual states when it emulates some operations of Target OS, such as interrupt/exception forwarding.

The Changes Of CPU State

- smsw: low 16 bits of CR0
- pushf/pushfd: IF, IOPL in EFLAGS
- popf/popfd: ditto

Segmentation Virtualization

- Ring Compression is the core of the whole virtualization scheme, and its implementation depends on the Segment Shadowing technique.
- Rather than allowing hardware use the Target descriptor table directly, Rootkit provide a shadow descriptor table (Shadow DT) instead.

Shadow DT Structure & Shadowing Algorithm

- Shadow DT Structure: shadow entries + 6 cached entries + some Rootkit reserved entries. All shadow entries are initialized to be not present with segment present bits (SPB) set to 0 (unshadowed state). Cached and Rootkit reserved entries's DPL set to 0, SPB set to 1.

- Shadowing Algorithm used when synchronizing descriptor pairs: For Data/Code segment descriptor, DPL should be modified from 0 to 1 (or 3), and limit be truncated if overlapped with Rookit space. For Gate descriptor, its target code segment selector value should be modified to 0

Segment Related Operations Virtualization

- Target Redefine DT
- Target Modify Descriptors
- Target Remap/Unmap DT
- Target Segment Loading
- Synchronize Segment A (Accessed) Bit

Inter-Segments Control Transfer Virtualization

- Task Switching
- Call Gate
- Direct jmp & call/ret Between Segments
- Interrupt/Exception
- sysenter/sysexit

The Changes Of Segment System

- The Change Of GDTR
- The Change Of GDT Descriptors
- The Change Of Segment Selector
- Segment Related Non-Virtualizable Instructions

The Segment Irreversibility Problem

- The deferred segment synchronization scheme ensures that cached descriptors would be updated timely when first loading (non-natural loading) of segment descriptor.

- Usually this problem only happens during system Boot phase when switching from real mode to protected mode.

Paging Virtualization

- Two reasons for shadow paging: 1) The stealth in both linear and physical address space. 2) Access and Protection bits used to play various virtualization tricks.
- Rather than allowing hardware use the Target page directory and page table directly, Rootkit provide a Shadow Page Table instead.

Shadow DT Structure & Shadowing Algorithm

- Initially the shadow page table is empty, that means all entries (PDE) except for the one that maps Rootkit itself are marked as invalid.
    - Target kernel mode is compressed to ring1.
    - Target's all modes are compressed to ring3.

The Working Procedure Of Shadow Paging

- When #PF occurs during Target execution, Rootkit should emulate hardware MMU’s action to traverse Target’s page table: If no valid mapping is found, then forwards and lets #PF to be handled by Target OS. When valid mapping is found and it’s not caused by physical trace (checking the fault address (CR2)), then begins the shadowing procedure.

- Shadows the corresponding PDE, creates a new page table and shadows corresponding PTE (marks other PTEs as not invalid), then writes protect the appropriate portion of Target table, finally re-execute the faulting instruction.

Physical Trace & Linear Trace

- Physical Tracing: Rootkit has the ability to install read, write, read/write traces on Target’s physical memory pages, and be notified when accesses (read or write) to these pages occur.
- Linear Tracing: A mechanism used by Rootkit to detect the mapping change (unmap or remap) of a given linear region range of Target OS.

Paging System Related Operations Virtualization

- Target Read PDBR
- Target Load PDBR
- Target Modify PDE/PTE
- Target Flush TLB
- Maintain The Accessed/Dirty Bits

Handle The Page Fault

- When #PF occurs during Target execution, Rootkit should emulate hardware MMU’s action to traverse Target’s page table: If no valid mapping is found, then forwards and lets #PF to be handled by Target OS. When valid mapping is found and it’s caused by physical trace, then cancel the trace installed on the mapping temporarily and restore mapping to its degradation state after single-stepping the faulting instruction.

Linear Address Space Confliction

- Relocate the Rookit itself when its occupied space is also mapped by Target. Just adjusts its code segment base, no memory copy is needed. Don’t forget to reload all processor states affected.

- As an optimization, maps the Rookit at a region which is known not used by a given Target OS at the first beginning.

The Changes Of Paging System

- The Change Of CR3
- The Change Of Page Directory Table/Page Table

TSS Virtualization

- The Virtualization of TSS is a precondition for interrupt/exception virtualization and I/O instructions trapping.
- Rootkit provides a private TSS, TSS descriptor, and a ring0 stack region.

TSS Related Operations Virtualization

- Target Read TR
- Target Load TR

Interrupt/Exception Virtualization

- The virtualization of Interrupt/Exception is a prerequisite. Firstly, it’s helpful to hide the virtualization fact. In addition, it’s the only approach and foundation for Rootkit to gain control and play all virtualization tricks.

- Rootkit provides a private IDT that pointed directly by hardware IDTR, and a single handler (trampoline) for each kind of exception, software interrupt, hardware interrupt.

Processing Interrupt/Exception

- Capture Interrupt/Exception

- Handle the Exceptions caused by virtualization directly

- Forward the hardware Interrupts, Software Interrupts and the Exceptions caused by Target OS Itself

- Interrupt/Exception returns of the Target OS

IDT Related Operations Virtualization

- Target Read IDTR
- Target Load IDTR

Device Virtualization

- Rootkit is not a virtual machine project, so it’s unnecessary to provide a complete set of virtual devices to Target OS, but this doesn’t mean that Rootkit has no ability to interpose and control the device related operations of Target OS.

Device Related Operations Virtualization

- Port Mapped I/O
- Memory Mapped I/O
- Hardware IRQ
- DMA

The Payload of VMBR

- Virtual Machine Introspection (VMI)
- Demonstration: Hide any process under icesword

Conclusion

- Not a perfect virtualization, but a proof of concept.
- Future works:
  - More Target OS supported.
  - Advanced Processor Features: APIC, SMP, Machine Check.
  - Dynamic Scanning.

Acknowledgement

- Firstly, to all forthgoers and their works.
- Next, to my colleagues and friends: Mr. Old Song, Mr. Anthony Bettini , and Miss Zang.
- Finally, to all people who have contributed to this presentation.

Primary Documents Referenced

- 【1】Intel Corporation 《Intel Architecture Software Developer’s Manual Volume 2》 1997
- 【2】Intel Corporation 《Intel Architecture Software Developer’s Manual Volume 3》 2003
- 【3】VMware Inc. 《System And Method For Virtualizing Computer Systems》 Dec. 2002
- 【4】VMware Inc. 《System And Method For Facilitating Context-Switching In a Multi-Context Computer System》 Sep. 2005
- 【5】VMware Inc. 《Deferred Shadowing of Segment Descriptors In a Virtual Machine Monitor For a Segmented Computer Architecture》 Aug. 2004
- 【6】VMware Inc. 《Dynamic Binary Translator With A System And Method For Updating And Maintaining Coherency Of A Translation Cache》 Mar. 2004
- 【7】VMware Inc. 《Method And System For Implementing Subroutine Calls And Returns In Binary Translation SubSystem Of Computers》 Mar. 2004
- 【8】John Scott Robin, Cynthia E. Irvine 《Analysis of the Intel Pentium’s Ability to Support a Secure Virtual Machine Monitor》 Aug. 2000
- 【9】K. Lawton 《Running Multiple Operating Systems Concurrently On an IA32 PC Using Virtualization Techniques》 1999

Primary Documents Referenced

- 【10】Samuel T. King, Peter M. Chen, Yi-Min Wang 《SubVirt Implementing Malware With Virtual Machines》 2006
- 【11】Jeremy Sugerman, Ganesh Venkitachalam, Beng-Hong Lim 《Virtualizing IO Devices on VMware Workstation’s Hosted Virtual Machine Monitor》 Jun. 2001
- 【12】Charles L. Coffing 《An x86 Protected Mode Virtual Machine Monitor for the MIT Exokernel》 May. 1999
- 【13】Prashanth P. Bungale, Swaroop Sridhar, Jonathan S. Shapiro 《Low-Complexity Dynamic Translation in VDebug》 Mar. 2004
- 【14】Prashanth P. Bungale, Swaroop Sridhar, Jonathan S. Shapiro 《Supervisor-Mode Virtualization for x86 in VDebug》 Mar. 2004
- 【15】Jack Lo 《VMware and CPU Virtualization Technology》 2005
- 【16】Dave Probert 《Windows Kernel Internals II Virtual Machine Architecture》Jul. 2004
- 【17】Steve McDowell, Geoffrey Strongin 《Virtualization Technology For AMD Architecture》
- 【18】Narendar B.Sahgal, Dion Rodgers 《Understanding Intel® Virtualization Technology (VT) 》

Thank For Attending

Question & Discussion Time