Overview of System Virtualization: The most powerful platform for program analysis and system security

Zhiqiang Lin
Department of Computer Science
University of Texas at Dallas

April 4th, 2012

Outline

1. Background
2. Virtualization Types
3. Key Techniques
4. Summary

Why virtualization?

- Why virtualization?
- Why not virtualize?
- Today's data center
- Cloud computing
- Fault resist
- Cost Effect
- Easy to maintain

CS 6V81-05: System Security and Malicious Code Analysis
### What is Virtualization

**Hardware**
- Virtual Machine Monitor

**Virtualization Properties**
- **Isolation**
  - Fault Isolation
  - Fundamental property of virtualization
  - Performance Isolation
  - Accomplished through scheduling and resource allocation
- **Encapsulation**
  - All VM state can be captured into a file
  - Operate on VM by operating on file
  - mv, cp, rm
- **Interposition**
  - Proportional to virtual HW model
  - Independent of guest software configuration

---

**Linux**
- Linux (devel)

**Operating Systems**
- XP
- Vista
- MacOS
Interposition

- All guest actions go through monitor
- Monitor can inspect, modify, deny operations
- Example:
  - Compression
  - Encryption
  - Profiling
  - Translation

Why Not the OS?

- It about interfaces
  - VMMs operate at the hardware interface
  - Hardware interface are typically smaller, better defined than software interfaces
- Disadvantages of being in the monitor
  - Low visibility into what the guest is doing

Virtualization benefits

- Increased resource utilization:
  - Server consolidation
  - Multiplexing
- Mobility
- Enhanced Security
- Test and Deployment
- ...
### Background

#### Virtualization Types

#### Key Techniques

#### Summary

### Mobility: load balance (Migration)

- **Server 1**
  - CPU Utilization = 90%

- **Server 2**
  - CPU Utilization = 50%

### Enhanced Security

- **Conventional VM**
- **Secure VM**

### Testing and Deployment

- **Develop**
- **QA**
- **Deploy**
Types of Virtualization

Process Virtualization
- Language construction
  - Java, .NET
- Cross-ISA emulation
  - Apple’s 68000-PowerPC to Intel X86 Transition
- Application virtualization
  - Sandboxing, mobility

System Virtualization
- VMware
- Xen
- Microsoft’s Viridian

Taxonomy

Process VMs
- Same ISA
- Dynamic Translators
  - Classic-System VMs
  - Whole-System VMs
- Dynamic Binary Optimizers
  - Hosted VMs
  - Co-designed VMs

System VMs
- Different ISA
- Multiple Systems
  - Hybrid VMs
  - HLL VMs

System Virtual Machine Monitor Architectures

- Traditional
  - Hosted
    - VMware Workstation
  - Hybrid
    - VMware ESX
    - Xen

Examples: IBM VM/370, Stanford DISCO
Hosted Virtual Machines

- **Goal:**
  - Run Virtual Machines as an application on an existing Operating System (QEMU, VirtualPC, VirtualBox, VMware-workstation)

- **Why**
  - Application continuity
  - Reuse existing device drivers
  - Leverage OS support
    - File system
    - CPU Scheduler
  - VM management platform

Hosted Monitor Architecture (VirtualBox)

Positives

- Installs like an application
  - No disk partitioning needed
  - Virtual disk is a file on host file system
  - No host reboot needed
- Runs like an application
  - Uses host schedulers

Negatives

- I/O path is slow
  - Requires world switch
  - Relies on host scheduling
  - May not be suitable for intensive VM workloads

Hosted Monitor Scheduling (VirtualBox)

Hosted Architecture Tradeoffs
<table>
<thead>
<tr>
<th>Background</th>
<th>Virtualization Types</th>
<th>Key Techniques</th>
<th>Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>VMware ESX 2.0</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Key Techniques
- **Para-virtualization**
  - Linux Guest
- **Hardware-supported virtualization**
  - Unmodified Windows
- **Isolated Device Drivers**

**Source:** Ottawa Linux Symposium 2006 presentation.
http://www.cl.cam.ac.uk/netos/papers/

### Hypervisor
- **Hardware-supported single-use monitor**
- **Characteristics**
  - Small size
  - Runs in a special hardware mode
  - Guest OS runs in normal privileged level
- **Uses**
  - Security
  - System management
  - Fault tolerance

**Outline**
1. Background
2. Virtualization Types
3. Key Techniques
4. Summary
### Key Techniques to implement Virtualization

- **Instruction Interpretation**
  - Emulate Fetch/Decode/Execute pipeline in software
  - **Positives**
    - Easy to implement
    - Minimal complexity
  - **Negatives**
    - Slow!

- **Binary Translation**
- **Trap-and-emulate**
- **Para-virtualization (hardware virtualization)**

### Example: CPUState

```c
static struct {
  uint32 GPR[16];
  uint32 LR;
  uint32 PC;
  int IE;
  int IRQ;
} CPUState;

void CPU_CLI(void) {
  CPUState.IE = 0;
}

void CPU_STI(void) {
  CPUState.IE = 1;
}
```

### Example: Virtualizing the Interrupt Flag w/ Instruction Interpreter

```c
void CPU_Run(void) {
  while (1) {
    inst = Fetch(CPUState.PC);
    CPUState.PC += 4;
    switch (inst) {
      case ADD:
        CPUState.GPR[rd] = GPR[rn] + GPR[rm];
        break;
      case CLI:
        CPU_CLI();
        break;
      case STI:
        CPU_STI();
        break;
      ...
      case INT:
        CPU_CLI();
        break;
      case CLI:
        CPU_CLI();
        break;
      case STI:
        CPU_STI();
        break;
      ...
      case X:
        CPU_X();
        break;
      case Y:
        CPU_Y();
        break;
      ...
    }
    if (CPUState.IRQ && CPUState.IE) {
      CPUState.IE = 0;
      CPU_Vector(EXC_INT);
    }
  }
}

void CPU_CLI(void) {
  CPUState.IE = 0;
}

void CPU_STI(void) {
  CPUState.IE = 1;
}

void CPU_Vector(int exc) {
  CPUState.LR = CPUState.PC;
  CPUState.PC = disTab[exc];
}
```
II. Binary Translator

VirtualizationTypes

KeyTechniques

Summary

Basic Blocks

GuestCode

Translator

TranslationCache

Callouts

CPUEmulationRoutines

TC

Index

vPC

mov ebx, eax
ccli
and ebx, ~0xfff
mov ebx, cr3
sti
ret

Guest OS + Applications

Unprivileged

Virtual Machine Monitor

Guest OS + Applications

Unprivileged

Virtual Machine Monitor

Page Fault

Undef Instr

vIRQ

MMU Emulation

CPU Emulation

I/O Emulation

Guest Code

Straight-line code

Basic Block

Control flow

III. Trap and Emulate

Guest Code

Translation Cache

start

mov ebx, eax

call HANDLE_CLI

and ebx, ~0xfff

mov ebx, cr3

sti

ret

mov [CO_ARG], ebx

call HANDLE_CR3

call HANDLE_STI

jmp HANDLE_RET

mov ebx, eax

call HANDLE_CLI

and ebx, ~0xfff

mov ebx, cr3

sti

ret
IV. Hardware virtualization, Para-virtualization (e.g., Xen)

- Modify operating systems to let OS to cooperate with VMM
  - Let OS runs in Ring1 and user apps run in Ring3
  - Change sensitive instructions to explicit calls to the VMM
  - No need to trap and emulate

Traditional Address Spaces

- \(0\) - \(4GB\)
  - Background Process
  - Operating System

- \(0\) - \(4GB\)
  - Current Process
  - Operating System

- \(0\) - \(4GB\)
  - RAM
  - Frame Buffer
  - Devices
  - ROM

Virtual Address Space

- \(0\) - \(4GB\)
  - Virtual Page Table

Physical Address Space

- \(0\) - \(4GB\)
  - Shadow Page Table

Shadow Page Tables

- \(0\) - \(4GB\)
  - Guest Page Table

- \(0\) - \(4GB\)
  - Guest Page Table

- \(0\) - \(4GB\)
  - Guest Page Table
Virtualization (or virtualisation), in computing, is the creation of a virtual (rather than actual) version of something, such as a hardware platform, operating system, storage device, or network resources.

**Impacts**
- Cloud computing
- Data centers
- Provides greater opportunities to security

**References**
- Xen and the art of virtualization
- Memory resource management for Vmware Esx Server
- http://labs.vmware.com/courseware (great virtualization resources)
- CPU Virtualization- Basic Virtualization
  http://labs.vmware.com/download/76/
- Memory Virtualization
  http://labs.vmware.com/download/77/