

Practical Taint-Based Protection using Demand Emulation

Alex Ho, Michael Fetterman,
Christopher Clark, Andrew Warfield,
& Steven Hand
Computer Laboratory, University of Cambridge



Internet is dangerous

*Internet is dangerous
worms, viruses, spyware, adware*

*Internet is dangerous
worms, viruses, spyware, adware
don't execute malicious code*

Overview

- Virtualization
 - fast
 - minimal interposition
- Emulation

Overview

- Virtualization
- Emulation
 - greater control
 - slower

Overview

- Virtualization
- Emulation
- Full-time protection

Overview

- Virtualization
- Emulation
- Full-time protection
 - Don't execute code from the Internet

Tainting

Tainting

- Why
 - Stack smashing attacks
 - Heap corruptions
 - Root toolkits
- What
- How

Tainting

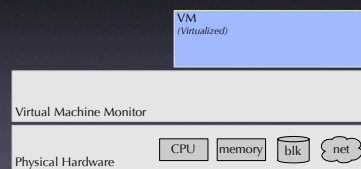
- Why
- What
 - Divide processor's registers into two classes: data & system
 - Untrusted I/O considered tainted
 - Prohibit system registers from referencing tainted data
- How

Tainting

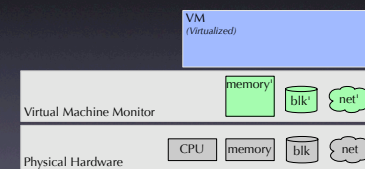
- Why
- What
- How
 - Memory: bit per byte
 - Processor: bit per register, track taint

Architecture

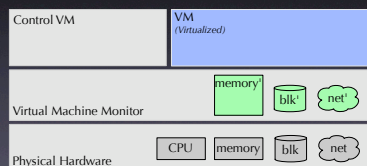
Virtualization



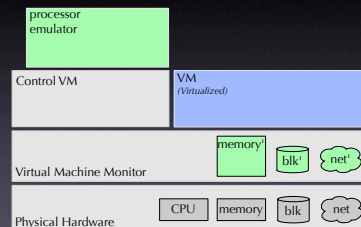
Virtualization



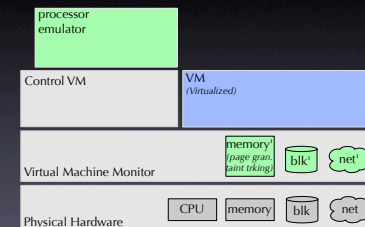
Virtualization



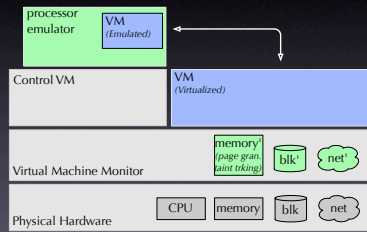
Virtualization & Emulation



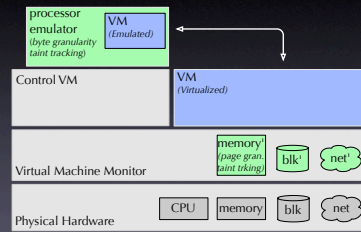
Virtualization & Emulation



Virtualization & Emulation

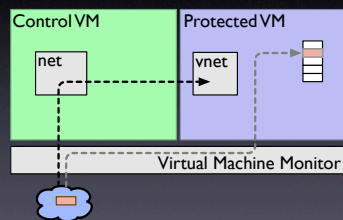


Virtualization & Emulation

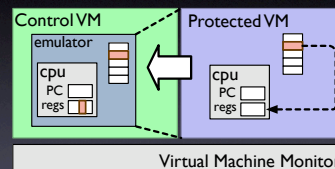


Example

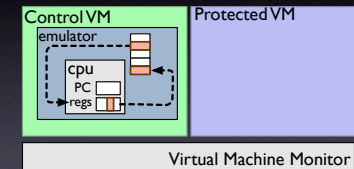
Packet Arrives



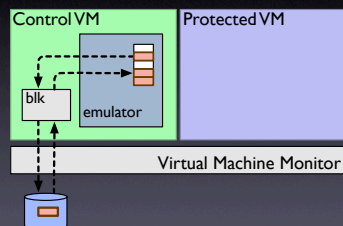
Protected VM accesses tainted data



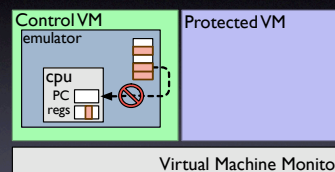
Emulator tracks tainted data



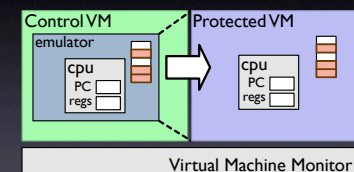
Disk I/O



VM attempts to execute tainted data

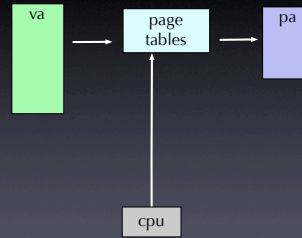


Normal execution resumes

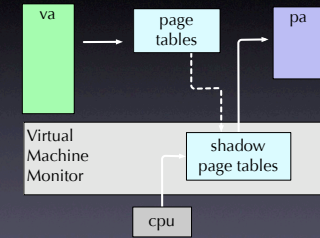


Details

Triggering the virtualization to emulation transition



Triggering the virtualization to emulation transition



Emulation

```
mov 0xc0379da8, %eax
```

x86

Emulation

```
mov 0xc0379da8, %eax
movl_A0_im 0xc0379da8
ldl_T0_A0
movl_EAX_T0
```

x86

micro ops

Emulation

```
mov 0xc0379da8, %eax
movl_A0_im 0xc0379da8
ldl_T0_A0
movl_EAX_T0
```

x86

micro ops

```
void op_movl_A0_im (u32_t imm) {
    A0 = (u32_t) imm;
}
void op_ldl_T0_A0 (void) {
    T0 = (u32_t)ldl_p((u8_t *)A0);
}
void op_movl_EAX_T0 (void) {
    EAX = T0;
}
```

emulator code

Emulation + Tainting

```
mov 0xc0379da8, %eax
movl_A0_im 0xc0379da8
ldl_T0_A0
movl_EAX_T0
```

x86

micro ops

```
void op_movl_A0_im (u32_t imm) {
    A0 = (u32_t) imm;
    taint_register(S_A0, 0);
}
void op_ldl_T0_A0 (void) {
    T0 = (u32_t)ldl_p((u8_t *)A0);
    taint_register(S_T0, check_memory(A0, 4));
}
void op_movl_EAX_T0 (void) {
    EAX = T0;
    taint_register(R_EAX, check_register(S_T0));
}
```

emulator + taint code

When do we revert to virtualization?

- After each instruction

When do we revert to virtualization?

- After each instruction
- When entering the virtual machine monitor

When do we revert to virtualization?

- After each instruction
- When entering the virtual machine monitor
- At the end of each translation block chain

When do we revert to virtualization?

- After each instruction
- When entering the virtual machine monitor
- At the end of each translation block chain
- Incorporate hysteresis

Performance

Performance

- Micro benchmarks
 - lmbench2 (1.1x ~ 3.7x)
 - fork (3.7x) (259 µsec)
 - fork + /bin/sh (1.2x) (6,724 µsec)
- Macro benchmark

Performance

- Micro benchmarks
- Macro benchmark
 - netcat (1.5x transmit, 18x receive)
 - ssh file transfer (132x transmit, 155x receive)
 - SPECweb99 (52.5x)

Lessons

- V2E & E2V transition cost
 - Engineering decision
 - Re-locate emulator
- False tainting
- Application awareness

Lessons

- V2E & E2V transition cost
- False tainting
 - Page granularity
 - ECC memory
- Application awareness

Lessons

- V2E & E2V transition cost
- False tainting
- Application awareness
 - Operating system changes
 - Memory classes

Summary

- Comprehensive & high performance
- Full-time protection
- V2E & E2V