

Controlling the Speed of Virtual Time for Malware Deactivation

Keisuke Okamura

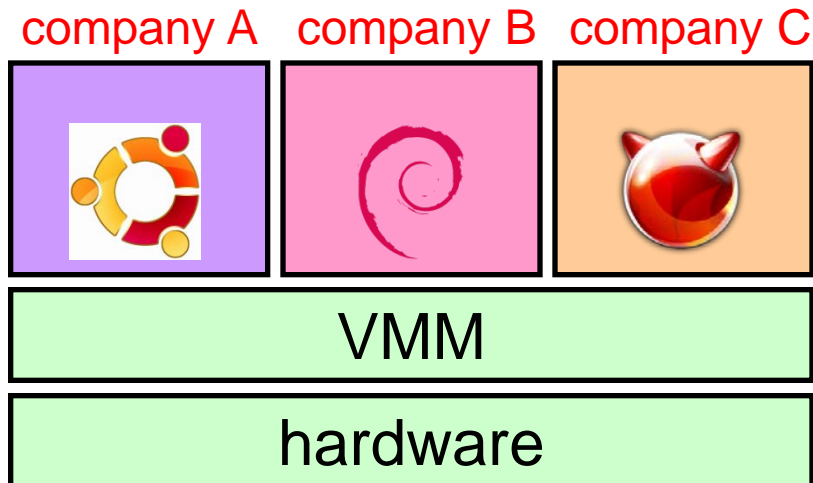
Yoshihiro Oyama

The University of Electro-Communications



Background

- IaaS (Infrastructure as a Service)
 - Ex.: Amazon EC2, Rackspace Cloud
 - IaaS provider lends VMs to customers
 - Customers freely use their own VMs

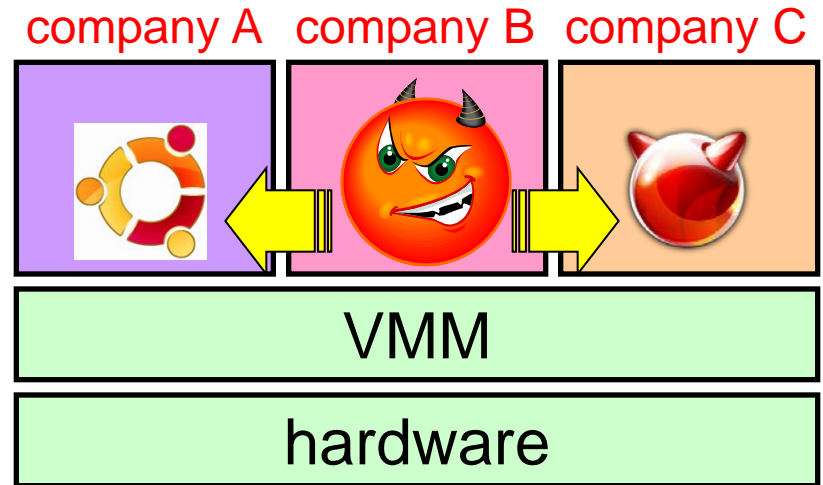


administrator of VMM \neq
administrator of guest OS

Malware Infection

- One guest OS is infected with malware that consumes resource

- Wastes CPU, memory, ...
- May have bad effect on other guest OSes



- Much literature showed that some malware could be detected from the VMM layer
 - VMwatcher [Jiang et al. '10], Lares [Brian et al. '08], Lycosid [Jones et al. '08]

Problem

- VMM administrator cannot modify the data managed by the guest OS
 - Ethically: Customer possesses the guest OS
 - Technically: Kind and version of the guest OS is unknown --- Windows? Linux? What version?
- Unfortunately, existing countermeasures are limited to coarse-grain ones
 - Ex.1: Stop whole VM
 - Ex.2: Drop all network packets from/to the VM
 - They affect even good processes running in the guest OS!

Goal

- Develop a VMM-based method for deactivating malware
 - Pinpointing the malware process
 - Mostly guest-OS-independent
 - Specifically, the method slows a malware process significantly
- Implement a system based on the method, HyperSlow, and demonstrate the effectiveness

HyperSlow

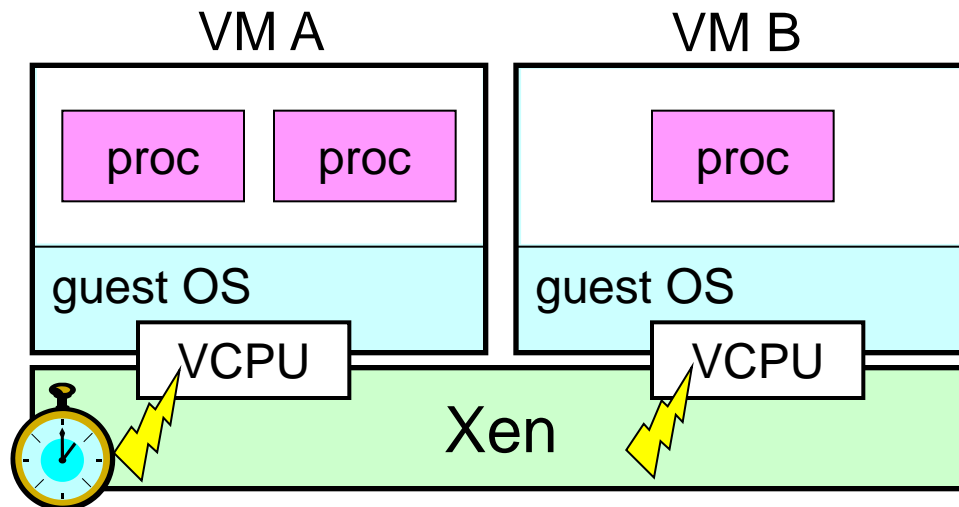
- Increases the speed of virtual time only while a malware process is running
 - It changes the rate of virtual timer interrupts and system time (elapsed time from boot)
- Is not a malware detection system, but a prevention system
 - It must cooperate with other detectors
- Xen-based

Process Scheduling Basics

- OS kernels (incl. Linux and Windows) use timer interrupts and/or system time for process scheduling
- Linux case:
 - When kernel receives a timer interrupt, it calculates the time consumed by the current process
 - Consumed time is calculated using system time
 - System time is calculated from hardware clock such as TSC
 - If the process has consumed all of the assigned time slice, it is preempted

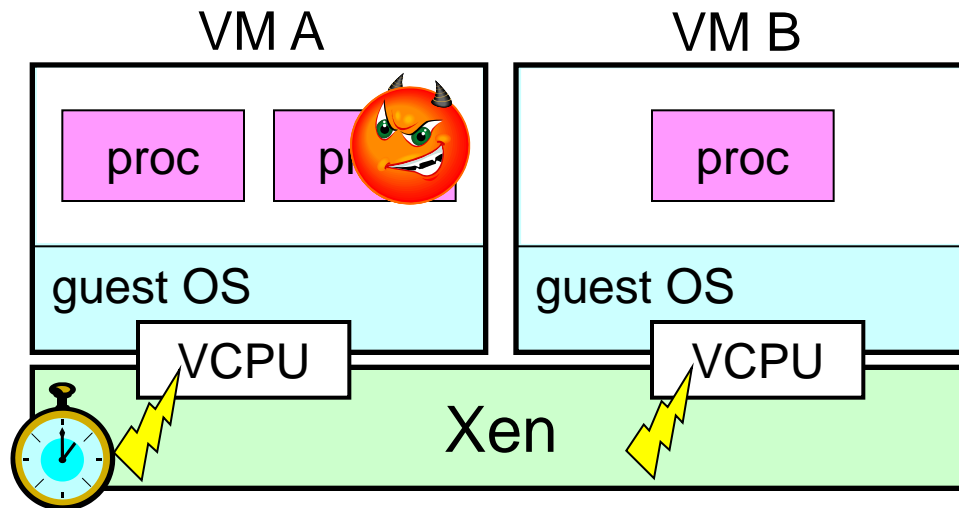
Virtual Timer Interrupts in Xen

- Xen periodically injects virtual timer interrupts to virtual CPUs
 - Based on them, guest OS schedules processes and performs timekeeping

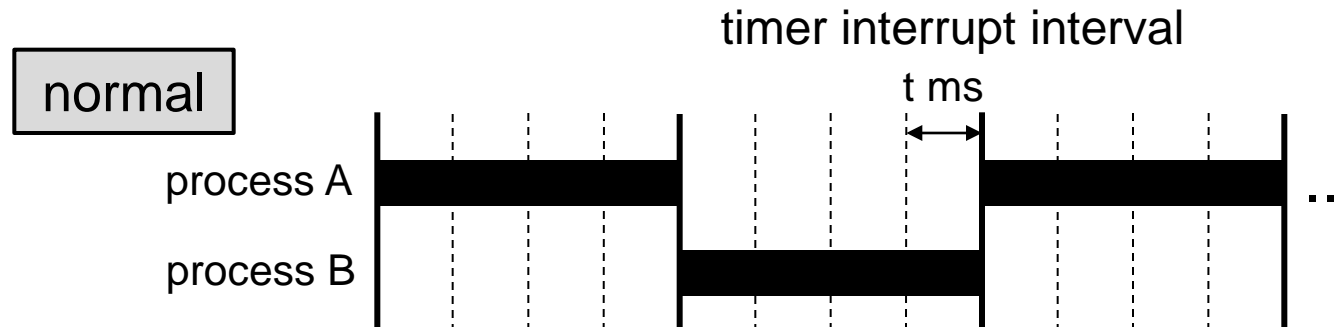


Basic Idea

- HyperSlow extremely shortens the interval between virtual timer interrupts, only while a malware process is scheduled
 - Promotes context switch by having guest OS kernel misunderstand the elapsed time
 - Uses CR3 register as a pseudo PID [Jones et al. '06]

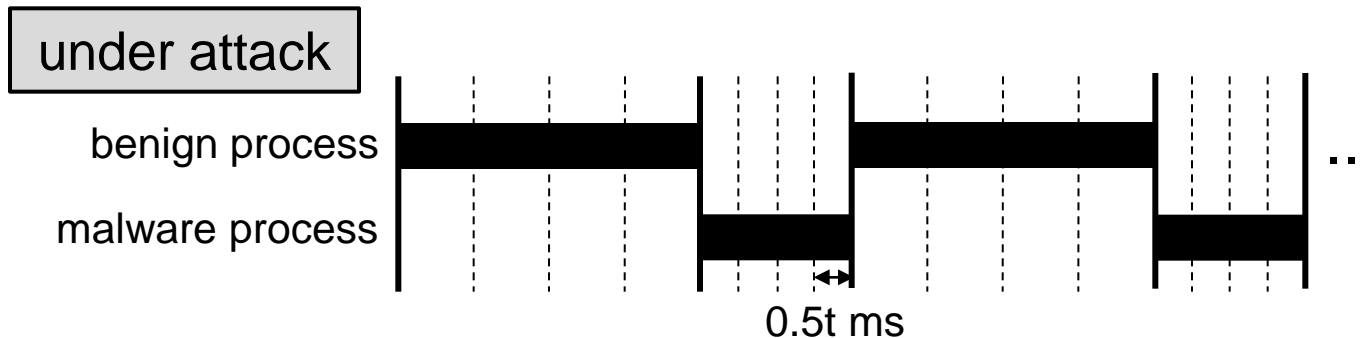


Example



While the malware is running, HyperSlow sets

- Timer interrupt interval: $\frac{1}{2}$
- Speed of system time: twice

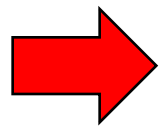
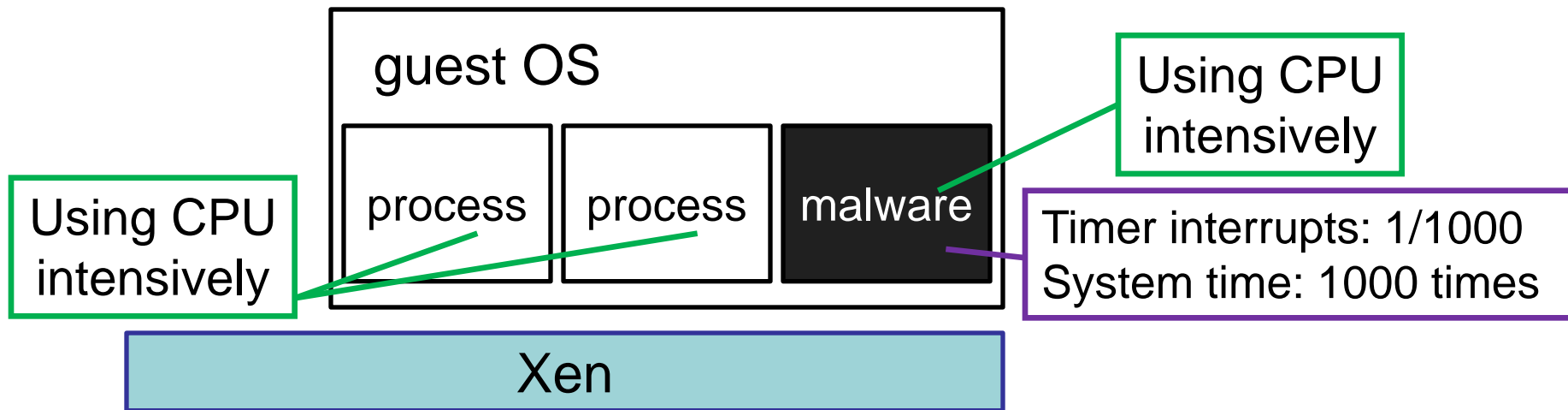


Malware obtains only half of the original time slice!

Experiments

- Platform
 - Intel Core 2 Duo 2.53 GHz, 2 GB memory
 - Xen 4.0.0
 - Guest OS:
 - Debian lenny (Linux 2.6.26), paravirtualized
 - Memory: 512 MB
 - Pinned to one physical CPU
- Three setting
 - Ideal setting
 - Web server and CPU-DoS malware
 - E-mailing malware

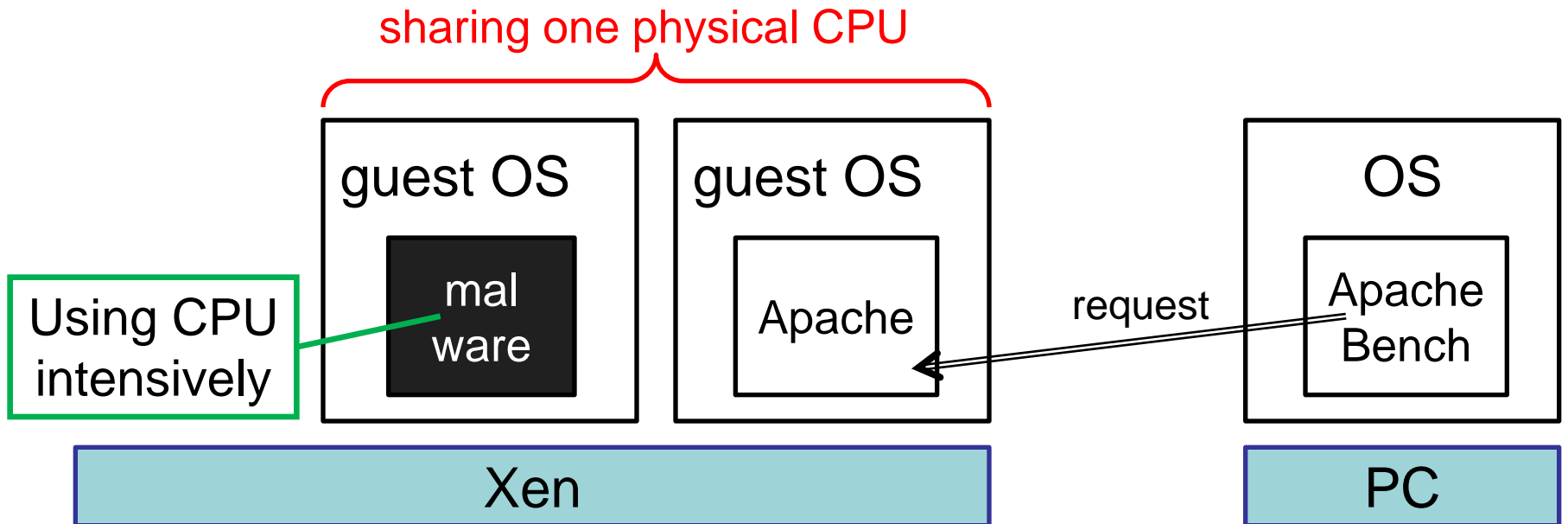
Ideal Setting



Speed of malware became 1/3869!
And, other processes were little affected!

- Why not 1/1000 but 1/3869?
-> Due to OS noises such as context switch?
Under investigation

Effect of CPU-DoS Malware on Web Server



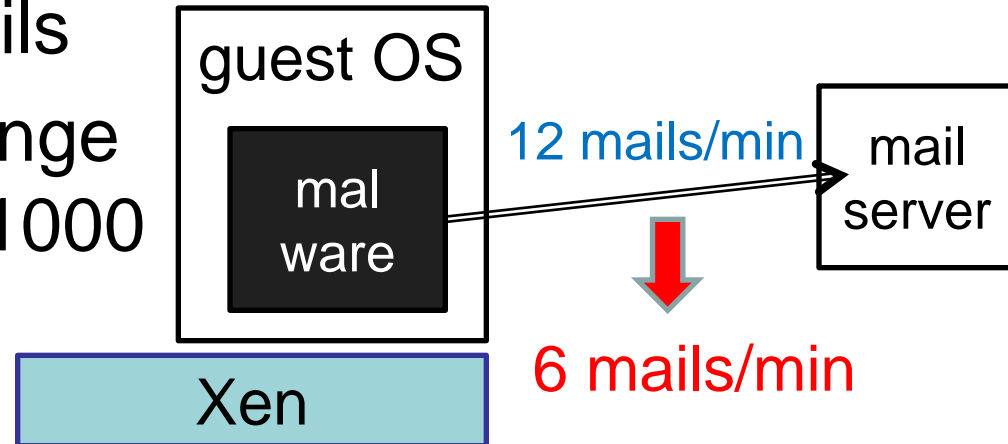
Effect of CPU-DoS Malware on Web Server

	average response time (ms)	worst response time (ms)
Original	8.52	3008
Malware added	76.63 (+799%)	46987 (+1462%)
HyperSlow added	11.92 (+39.8%)	9004 (+199%)

Greatly reduced negative effect on another guest OS

Slowing E-mailing Malware

- Malware sends e-mails
- We attempted to change malware speed to 1/1000



- Result:

HyperSlow could change the speed to approximately the half of the original

- Note that e-mailing software is I/O-intensive and/or sleeping most of time

Related Work

- FoxyTechnique [Yamada et al. VEE2007]
 - VMM-based system for changing resource management policy by OS kernel
 - No mention about slowing execution
- FoxyLargo [Yoshida et al. IWVT2008]
 - VMM-based system for slowing VCPU speed
 - Granularity is the whole OS, not a process
- Lycosid [Jones et al. VEE2008]
 - VMM increases CPU time consumed by a particular process
 - Focus on hidden process detection, not prevention

Conclusion and Future Work

- We proposed a malware deactivation method
 - Mostly OS-independent
 - Because it changes only the behavior of virtual hardware
 - Process granularity
 - It cannot completely stop malware, but will be useful for the first mild countermeasure in IaaS context
- Future work
 - Investigate side effects on timekeeping in guest OS
 - Combine with malware detection system
 - Do experiments on other platforms such as Windows, full-virtualized VMs, and other hypervisors