



REVIRT: ENABLING INTRUSION ANALYSIS THROUGH VIRTUAL- MACHINE LOGGING AND REPLAY

G. Dunlap, S. King, S. Cinar, M. Basrai, P. Chen
Department of Electrical Engineering and Computer Science
University of Michigan

Proceedings of Fifth USENIX Symposium on Operating Systems Design and Implementation, 2002

Presenter: Wilson Fung
Date: October 23, 2006

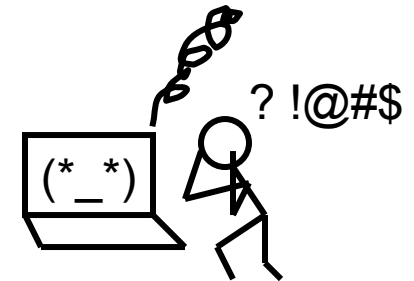


Introduction

- ReVirt: Logging with Virtual-Machine
 - Can replay long-term execution on VM instruction by instruction!
 - Overheads:
 - Imperceptible for CPU-bound workload
 - 13-58% for kernel-intensive workload
 - +0-8% for logging

Problem with current system logger

- Done by the kernel – buggy...
- Integrity
 - Attacker can forge/erase logs on a compromised system
- Completeness
 - Logged info is not sufficient to determine what exactly has happened (administrator can only guess ☹)



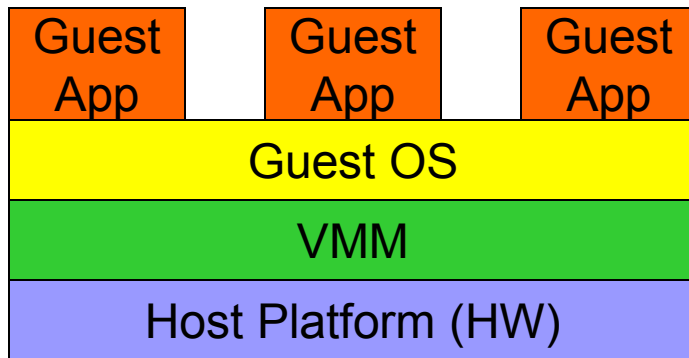
PARADOX: CURRENT LOGGERS FAILS WHEN THE KERNEL IS COMPROMISED...WHEN IT IS MOST NEEDED!



Virtual machines to the rescue!

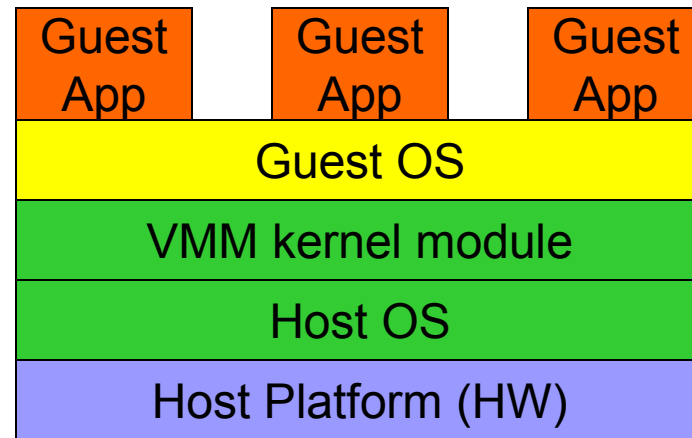
- Virtual-machine monitor (VMM):
 - A layer of software emulating the hardware of a complete system.
 - The guest OS has no direct access to real hardware, so is the attacker!
- Can VMM be compromised?
 - Yes, but much harder, as it has less code and provides a narrower interface.

2 ways to setup VMM



■ Direct-on-Host

- Less overhead
- Harder to log and replay (More on this)



■ OS-on-OS

- More overhead
- Limited interface allow easier logging and replay



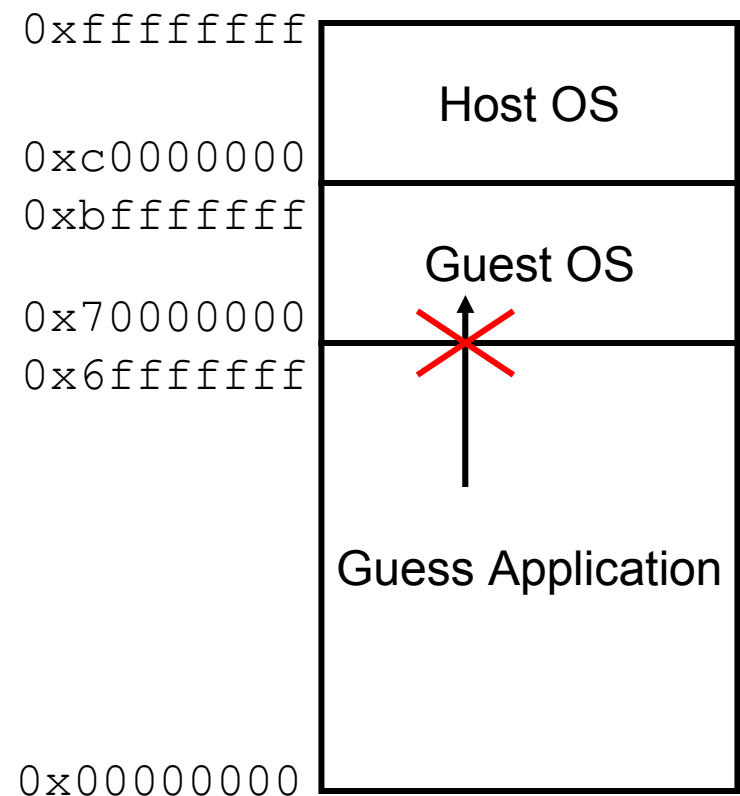
OS-on-OS VMM

■ UMLinux

- Guest OS version of Linux 2.4.18
- A user process running on Linux 2.4.18
- Communicate with host kernel via signal and system call to the VMM kernel module

OS-on-OS VMM (2)

- Memory accesses translated by host's MMU
 - UMLinux's guest kernel occupies a different address space.
 - Host memory protection are used to prevent guest application from accessing guest kernel's address space.



OS-on-OS VMM (3)

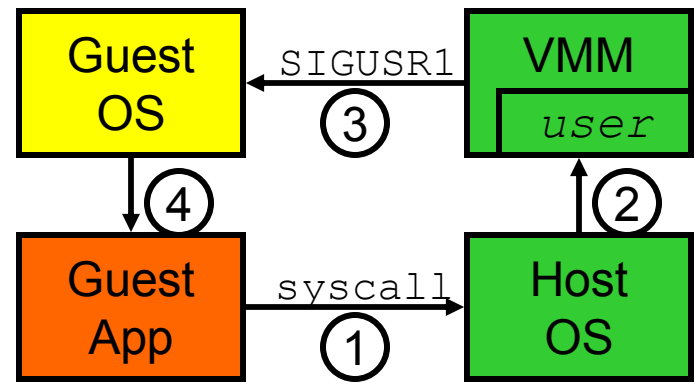
- Host components are emulated in UMLinux

Basically, all interrupts are emulated using signals

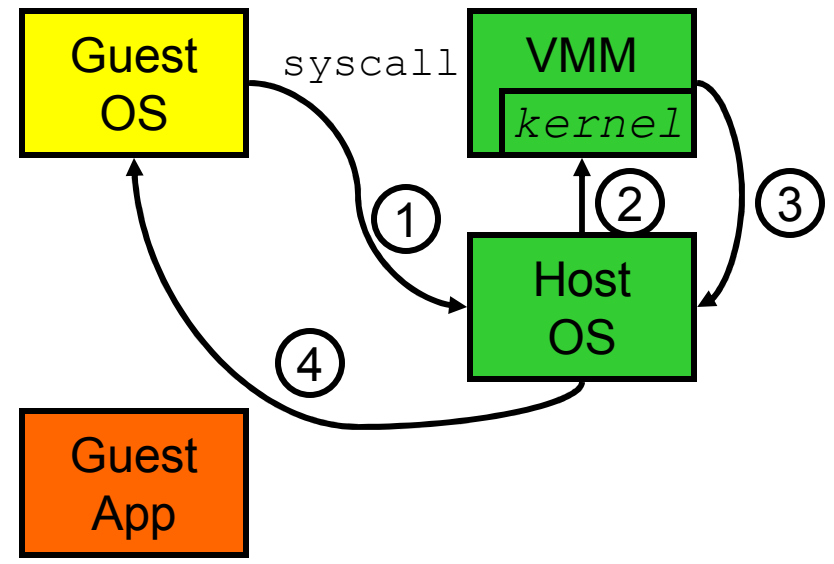
| | |
|---------------------------|---------------------------------|
| Hard disk | Host file or raw partition |
| CD-ROM/Floppy | Host dev/... |
| Network card | TUN/TAP virtual Ethernet Device |
| Console | TCP to host application |
| Video card | Remote X server |
| Current privilege level | VMM variable |
| System call | SIGUSR1 signal |
| Timer interrupts | Timer + SIGALRM signal |
| I/O device interrupts | SIGIO signal |
| Memory exception | SEGV signal |
| Enable/disable interrupts | Mask signals |

OS-on-OS VMM (4)

- System call from guest application

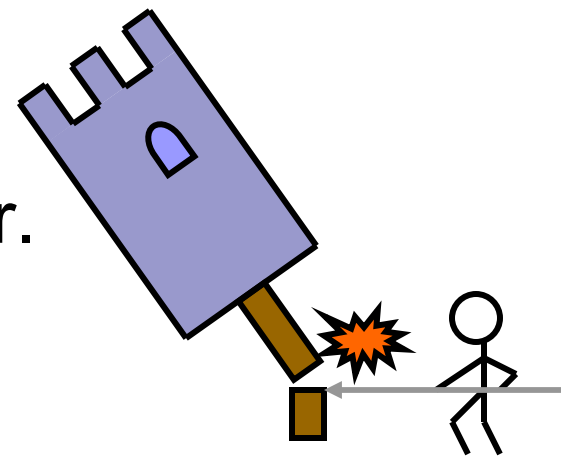


- System call from guest OS



Trusted Computing Base

- Trusted Computing Base (TCB) = VMM kernel module + Host OS
 - This part need to be secure for logging to work (ie. We trust this part of the system!)
- But, is it secure?
 - Depends...
 - OS-on-OS seems to be better.





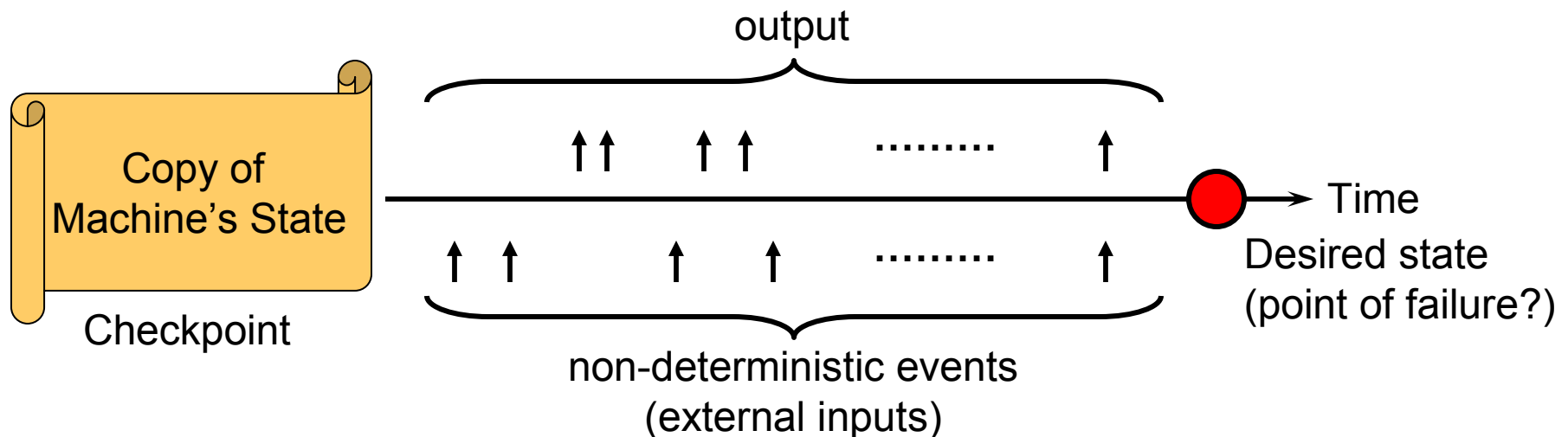
TCB for UMLinux (OS-on-OS)

- OS-on-OS is more secure because:
 - Even if the guest OS is compromised, attacker is only controlling a single host process.
 - Guest OS can access only a few host files/devices
 - VMM limits system calls available to guest OS (only 7% available)
 - Network stack on host OS is less exposed, as most network traffic to VM is processed by guest OS's transport level stacks.
 - But network traffic to host process still travels the whole network stack...

IT IS SAFER THAN DIRECT-ON-HOST SYSTEM.

Logging and Replay Overview

- Virtual Machine is a finite-state machine:
 - Given an initial state (checkpoint) and a set of timed input sequences (non-deterministic events), the state transition will be deterministic (can be replayed).





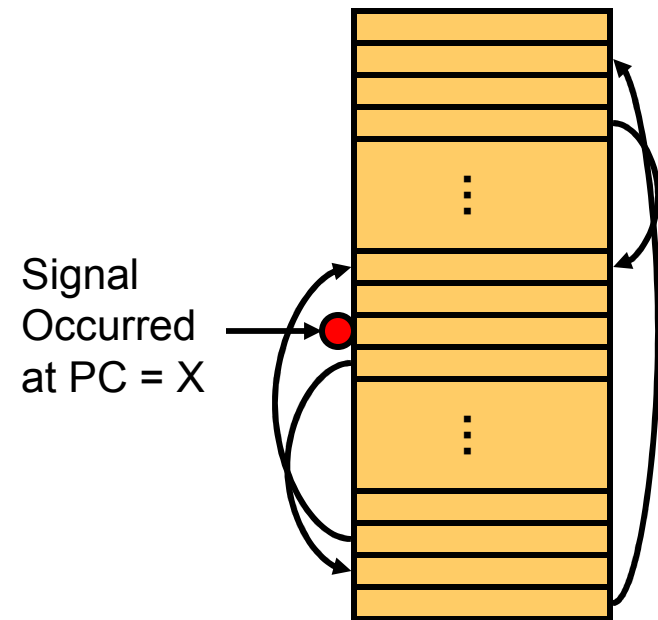
ReVirt: Logging

- Logging to enable replay of UMLinux running on x86 processor
 - Checkpoint → copy of virtual disk when VM is off
 - Later version allow checkpoint when VM is running by including CPU registers, VM's physical memory, and any state in the VMM or host OS that affects the VM.
 - Non-deterministic event → Asynchronous virtual interrupts (signals) and all input from external entities
 - Eg. keyboard, mouse, network packets, real-time clock, external drives
 - No instructions with non-deterministic results.
 - Cooperative Logging reduces log size between computers.
 - Problem: How to track the exact instruction at an interrupt?

ReVirt: Logging (2)

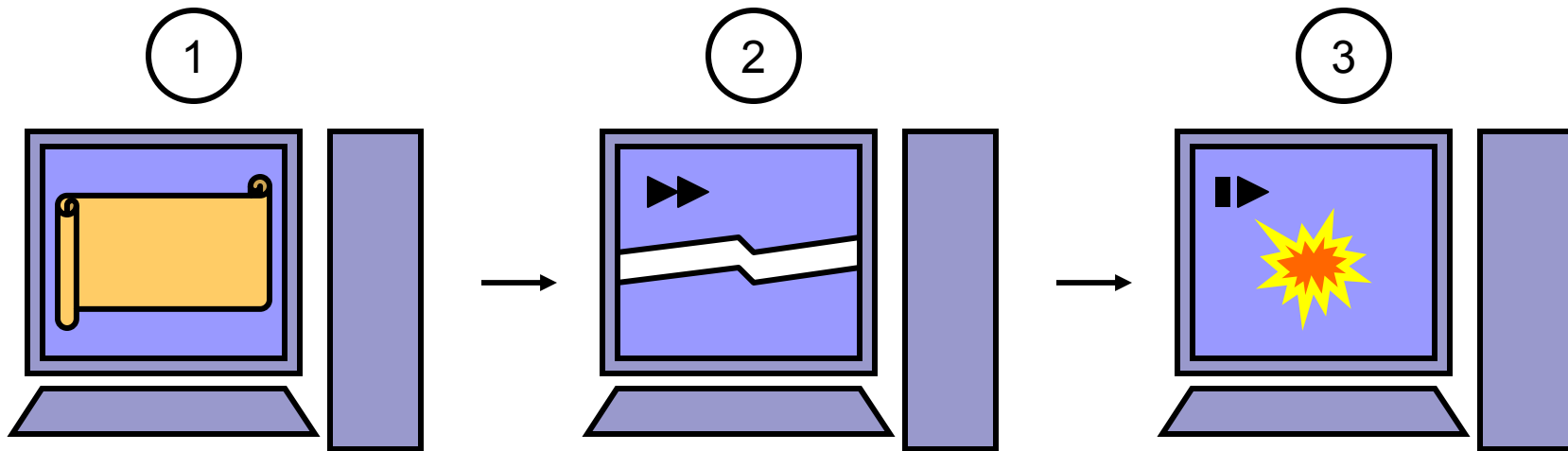
- During replay, signals need to be re-delivered at the same point of execution.
 - Log *program counter* (PC) and *branch_retired* for each signal.
 - PC pin-points to the exact instruction.
 - *branch_retired* records the VM's program flow (branches, hw interrupts, faults and traps).
 - Why not *retired_insns*?
 - Replay performance...

VM's Address Space



branch_retired defines which path is taken to get to PC = X

ReVirt: Replay



- Load Checkpoint
 - Initialized the VM's state

- First Phase
 - ReVirt configures `branch_retired` performance counter to generate interrupt when it is within 128 of the target `branch_retired`.
 - "Fast-Forward" VM

- Second Phase
 - Breakpoint at PC = target PC
 - At each breakpoint, `branch_retired` is compared to target `branch_retired`.





ReVirt: Attack Analysis

- ReVirt allows the administrator to replay the complete execution before, during, and after the attack.
 - Feature 1: Continue live execution at any point in the replay
 - Problem 1: Cannot go back to replay mode without checkpoint.
 - Feature 2: External debugging tool running on Host OS. Attacker cannot compromise these tools.
 - Problem 2: X Server, running outside VM, is not the same during replay! Different packets expected. Current solution is to a layer of TCP proxy.



Experiments

- Test configuration

- Hardware

- AMD Athlon 1800+
 - 256MB RAM
 - Samsung SV4084 IDE disk

- OS

- Host: Linux 2.4.18
 - Guest: Linux 2.4.18 (UMLinux) with 192MB RAM
 - Virtual hard disk on raw disk partition

Experiments: Virtualization Cost

| Workload | UMLinux runtime (normalized to host Linux) |
|------------------|---|
| POV-Ray | 1.01 |
| Kernel-build | 1.58 |
| NFS Kernel-build | 1.44 |
| SPECweb99 | 1.13 |
| Daily use | ~1 |

VMware: 1.25, UMLinux (original): 26, User-Mode Linux: 14



Experiments: Correctness

- 2 micro-benchmarks – PASSED
 - Two guest process with shared memory
 - Incrementing a variable in infinite loop
- Macro-benchmark – PASSED
 - Boot computer, start GNOME window manager, open several interactive terminals, and concurrently built two application on remote NFS server.

Experiments: Logging Overhead

| Workload | Logging runtime (wrt without logging) | Log growth rate | Replay runtime (wrt with logging) |
|------------------|--|-----------------|--------------------------------------|
| POV-Ray | 1.00 | 0.04 GB/day | 1.01 |
| Kernel-build | 1.08 | 0.08 GB/day | 1.02 |
| NFS Kernel-build | 1.07 | 1.2 GB/day | 1.03 |
| SPECweb99 | 1.04 | 1.4 GB/day | 0.08 |
| Daily use | ~1 | 0.2 GB/day | 0.03 |



That's slow!



Experiments: Attack Analysis

- Simulated attack on Ptrace race condition
 - Known vulnerability
 - Non-deterministic
 - Trojan horse to /bin/l`s`
 - Backdoor to /etc/inetd.conf
- Each attack is replayed, identified and examined.
- No real in-field test ☹️



Conclusion

- ReVirt: CCTV for computers!
 - It can replay a instruction-by-instruction execution sequence of a computer system
 - Useful in determining and fixing the damage inflicted by intruders
- Capture non-deterministic events
 - Debugger for race conditions
- Reasonable overhead
 - 13-58% for kernel intensive application
 - 0-8% logging overhead



Recent Progress

- BackTracker

- Automatic attack analyzer based on ReVirt

- Gdb extension

- Reverse debugger with logging and replay
- Detect transient dual processor faults