ReVirt

Virtual machine fault-tolerance for logging and replay.

Eugene Chou (euchou@ucsc.edu)

Outline

- Motivation
- Background
- Design
- Capabilities
- Experiments
- Takeaways and discussion

Deploy now, fix later

- Hard to make code robust against all attack vectors.
- Instead, do analysis after compromise and try to fix code.
- Logs are needed to perform said analysis.
- Current system loggers lack:
  - Integrity
  - Completeness

Integrity

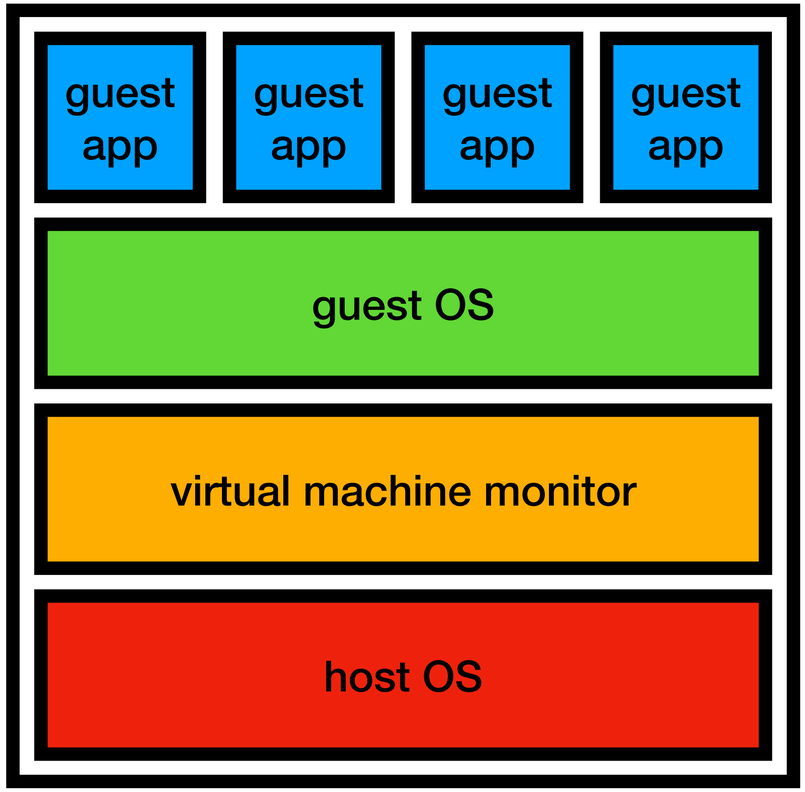
- Kernel is assumed to be trustworthy (even if it isn’t).
- Locally-stored logs: can be tampered with.
- Remotely-stored logs: can be falsified.
- Logs are intended to be used when system has been compromised.
  - And yet it assumes the system is not.

Completeness

- Typically not enough information stored for analysis.
- Can’t accurately replay without knowing exactly what happened.
- Deterministic and non-deterministic events should be stored.
  - Bugs tend to propagate from non-deterministic behavior.
  - E.g. time-of-use race conditions.

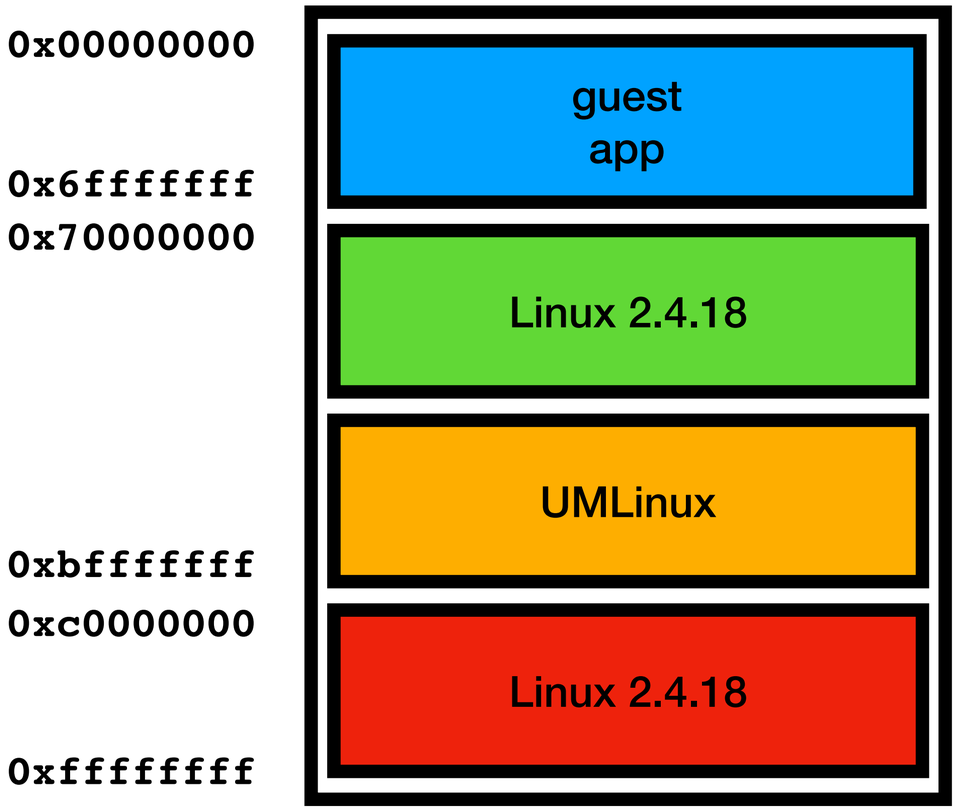
Goals and approach

- Log with integrity and completeness.
- Run target OS as guest OS in a virtual machine.
  - Separate logger domain from target OS domain.
- Works even if target OS is compromised to begin with.
- Need to trust virtual machine monitor (VMM).



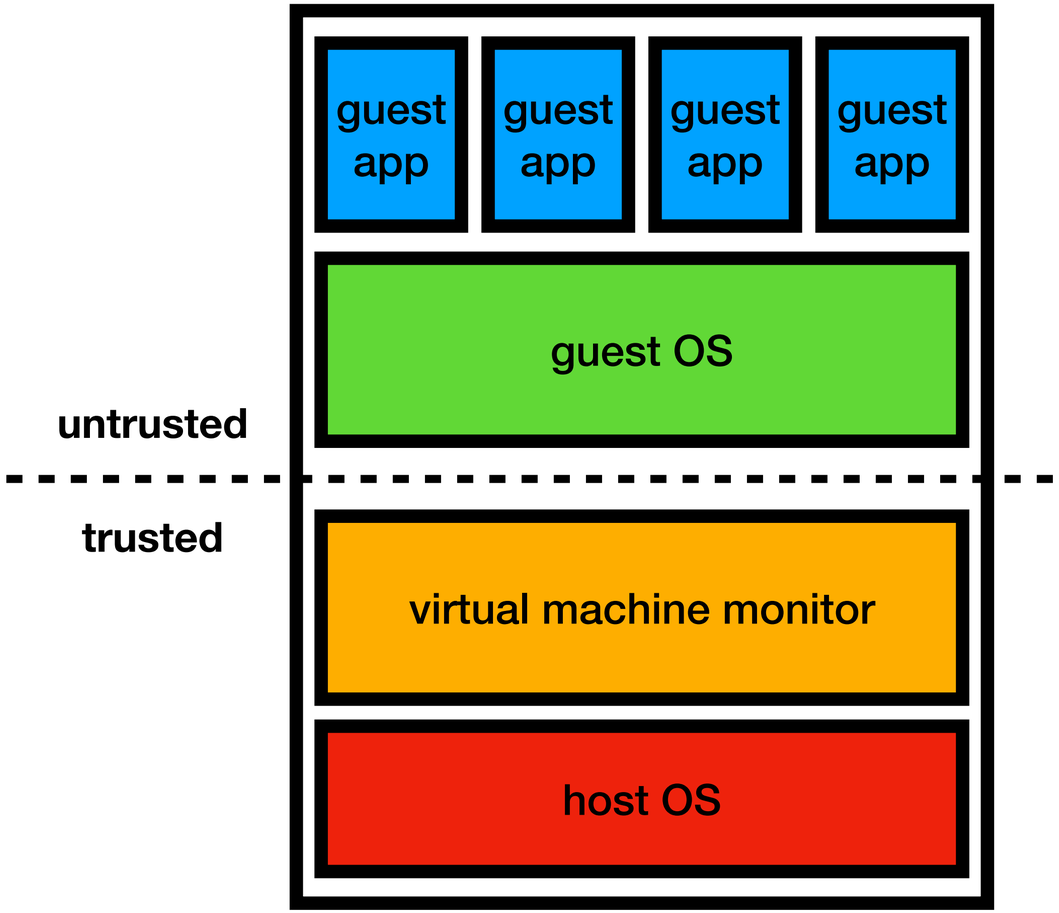
ReVirt and the UMLinux VM

- ReVirt is a set of modifications to some host.
- UMLinux runs as a single process on the host.
- VMM is called before/after every syscall and signal.
- **OS-on-OS:** guest OS runs on host OS.
  - Guest OS drivers use host system calls and signals.
- **Direct-on-OS:** target apps run on host OS.



What is trusted?

- **Trusted Computing Base (TCB):** all hardware/ software that is critical to the safety of some application.
- UMLinux TCB:
    1. VMM kernel module
    2. Host OS
- **Direct-on-host:** attacker can use host OS functionality freely.
- **OS-on-OS:** attacker has limited functionality in guest OS.
    - Not as much impact on host OS.



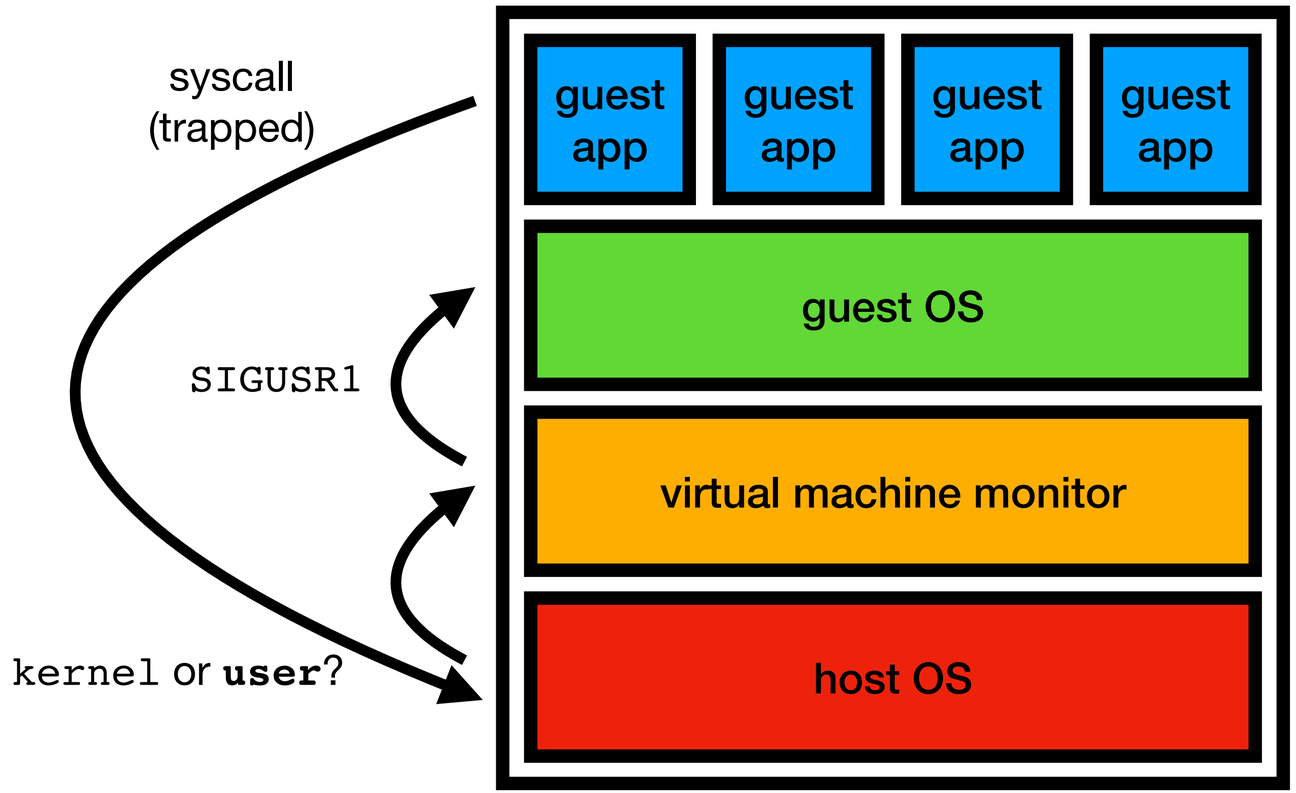
Why OS-on-OS?

- Direct-on-host not as secure.
  - Also harder to log/replay all host processes.
  - Easier to deal with one host process (the VM).
- Hard to replay scheduling between host processes.
  - Kernel doesn’t execute deterministically.
  - Affects instruction counts.
- Weaver et al. - *Non-Determinism and Overcount on Modern Performance Counter Implementations*

```
$ sudo perf stat -e instructions:k,instructions:u,instructions sleep 1

 Performance counter stats for 'sleep 1':

          583,229      instructions:k
          228,073      instructions:u
          811,632      instructions

       1.001835384 seconds time elapsed

       0.001682000 seconds user
       0.000000000 seconds sys


$ sudo perf stat -e instructions:k,instructions:u,instructions sleep 1

 Performance counter stats for 'sleep 1':

          572,842      instructions:k
          228,073      instructions:u
          811,632      instructions

       1.001794951 seconds time elapsed

       0.000000000 seconds user
       0.001687000 seconds sys
```

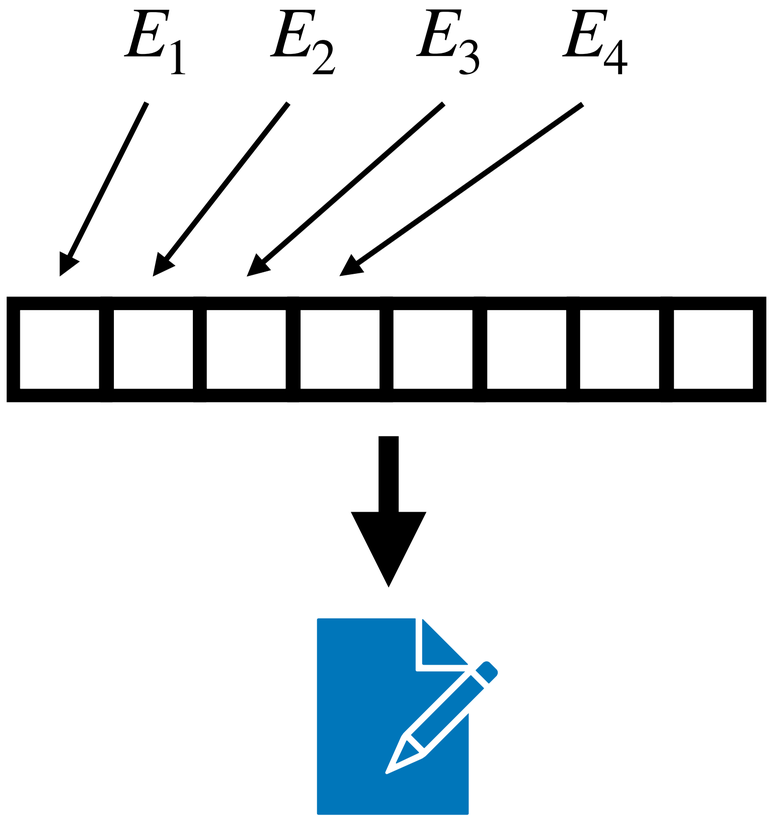
Privileges

- VMM must distinguish guest application and guest kernel syscalls.
- Track virtual privilege level in VMM.
  - Set to `kernel` for guest kernel.
  - Set to `user` for guest application.
- Guest application syscall redirected to guest kernel syscall trap handler.
- Guest kernel syscall is verified and passed to host kernel.



Logging in ReVirt

- Logging used for state recovery, and for replay.
  - Initial state is recorded (checkpoint).
- Logs stored in circular buffer in host kernel memory.
  - Periodically written to log file by a daemon (`rlogd`).
- Some deterministic events aren't logged.
- Most non-deterministic events are logged.
  - Interrupts, network activity, user input, etc.

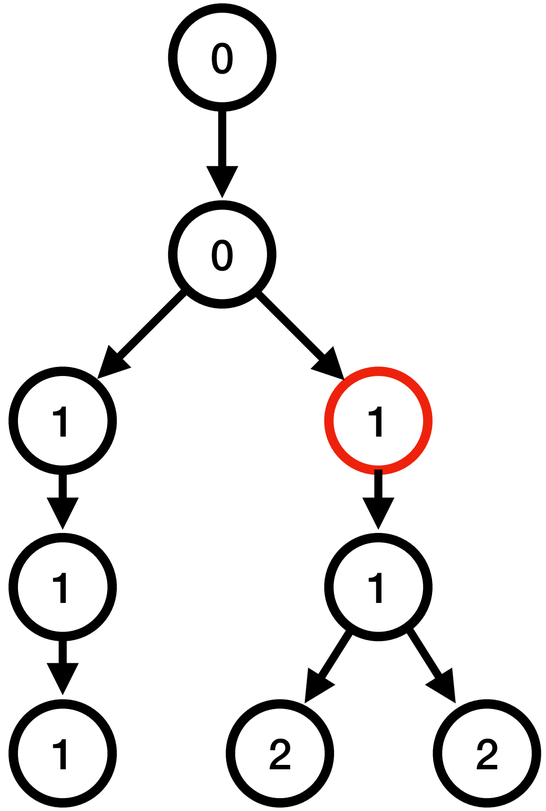


Logging non-deterministic events

- Non-deterministic events that don’t affect VM execution aren’t logged.
  - E.g. some host hardware interrupts, host process scheduling.
- Must log input from external entities and async virtual interrupts.
- **Asynchronous interrupt:** interrupts created at arbitrary times during execution, usually by external hardware.
- ReVirt logs program counter and number of branches since last interrupt.

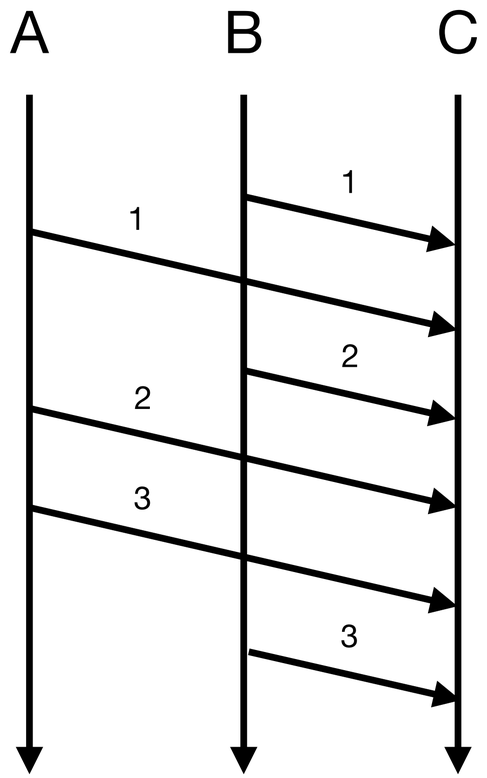
Replay interrupt delivery

- New async virtual interrupts blocked during replay.
- Old async virtual interrupts played back like they occurred during logging.
- To find instruction to deliver async virtual interrupt:
  - **Phase 1**
    - Generate interrupt after most branches in scheduling interval (modify `branch_retired`).
  - **Phase 2**
    - Set breakpoint on target instruction.
    - Compare current number of branches with expected amount.



Cooperative logging

- A lot of log data may go to network messaging.
- One computer's send is another's receive.
- Don't log data on cooperating receiver.
  - Just log identity of sending computer and log sequence number.
- Not yet implemented in ReVirt.



C's Log

(B, 1)
(A, 1)
(B, 2)
(A, 2)
(A, 3)
(B, 3)

Experiments

- ReVirt allows admins to switch to live execution during replay.
  - Can’t hop back into replay afterwards.
- Experiments ran on 5-workloads:
  - **POV-Ray:** ray-tracing program
  - **kernel-build:** compiles Linux 2.4.18 kernel
  - **NFS kernel-build:** compiles Linux 2.4.18 kernel and stores it on NFS server.
  - **SPECweb99:** benchmark used on 2.0.36 Apache web server
  - **Daily use:** email, editing, word processing, web browsing.

Virtual machine overhead

- Little overhead for interactive tasks.

- Kernel builds take longer due to syscalls.
  - Requires trap by VMM and signal to guest kernel.

Workload	UMLinux runtime (normalized to direct-on-host)
POV-Ray	1.01
kernel-build	1.58
NFS kernel-build	1.44
SPECweb99	1.13
daily use	≈ 1

Dunlap et al. - ReVirt: Enabling Intrusion analysis through Virtual-Machine Logging and Replay

Replay correctness

- Does the replay match up exactly with what happened?
- Error checking added to monitor any deviations from original execution.
  - All register values and `branch_retired` counter logged.
- Micro-benchmarks (each run 5 times):
  1. Two processes increment a shared variable and print its value.
  2. A process increments a variable in a loop, printing the value on interrupt.
- Macro-bechmark:
  - Boot computer, start window manager, build two applications on remote NFS server.

Time and space overhead

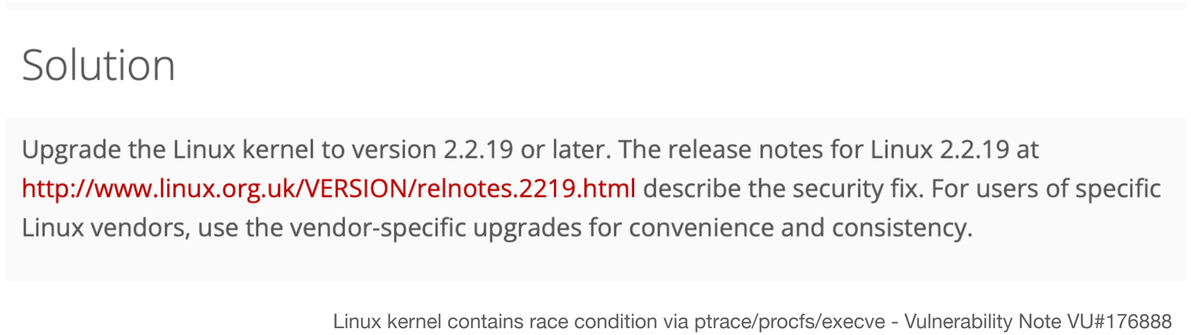
- Mostly deterministic workloads generate small logs (POV-ray).
- NFS kernel-build and SPECweb99 require logging of network packets.
  - Would perform like kernel-build with cooperative logging.
- Replay time occasionally faster than original runtime.
  - Idle periods are skipped.

Workload	Runtime with logging (normalized to UMLinux *without* logging)	Log growth rate	Replay runtime (normalized to UMLinux *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	0.88
daily use	≈ 1	0.2 GB/day	0.03

Dunlap et al. - ReVirt: Enabling Intrusion analysis through Virtual-Machine Logging and Replay

Analyzing a real attack

- Log and replay `ptrace` race condition in Linux kernel < 2.2.19.
  - Non-deterministic attack.
- `ptrace` allows process to attack to another and modify execution state and memory.
- Race condition exploited to attack to `setuid` and gain elevated privilege.



Solution

Upgrade the Linux kernel to version 2.2.19 or later. The release notes for Linux 2.2.19 at http://www.linux.org.uk/VERSION/relnotes.2219.html describe the security fix. For users of specific Linux vendors, use the vendor-specific upgrades for convenience and consistency.

Linux kernel contains race condition via ptrace/procfs/execve - Vulnerability Note VU#176888

Takeaways

- Use logging to record execution and enable playback.
- Decouple logger from target OS domain.
- Record non-deterministic events to replay non-deterministic attacks.
- Need to have some level of trust (TCB) in VMM and host OS.

Discussion

- Is ReVirt really suitable for use day-to-day use?
  - Its use-case seems to be more niche.
- Is there a minimum size to the trusted computing base?
  - What things *must* you be able to trust?
- Could an attacker produce a process with so much non-deterministic event-logging that the host system runs out of log space?