+

Cardinal Pill Testing of System Virtual Machines

http://steel.isi.edu/Projects/cardinal/

Hao Shi
haoshi@usc.edu
Abdulla Alwabel
alwabel@isi.edu
Jelena Mirkovic
mirkovic@isi.edu

Information Science Institute

University of Southern California
[USC/ISI]

USENIX Security 2014
San Diego, CA, USA

- Motivations
- Our Goal
- Related Work
- Architecture
- Evaluation
- Fixing Pills

Information Sciences Institute
USC School of Engineering

USC University of Southern California

+ Motivation

- Use of Virtual Machines in Current Malware Analysis
  - QEMU, Bochs, Anubis, TEMU, Ether, VMware, …
- Malware's Evasion Strategy (Anti-VM)
  - Detect VMs and change behavior from malicious to benign
    - **CPU semantic attacks**
      - Different execution of same instruction in a VM and a physical machine
    - Timing attacks, String attacks
  - Increasing popularity of anti-VM behavior in malicious binaries
    - 2.7% of 6,222 [Chen08], 25.6% of 1,686 [Lindofer11], 81.4% of 4,000,000 samples [Branco12]
- Definition of Pill
  - Combination of instruction mnemonic + register/memory parameter (ranges) that leads to different execution in a VM and a physical machine

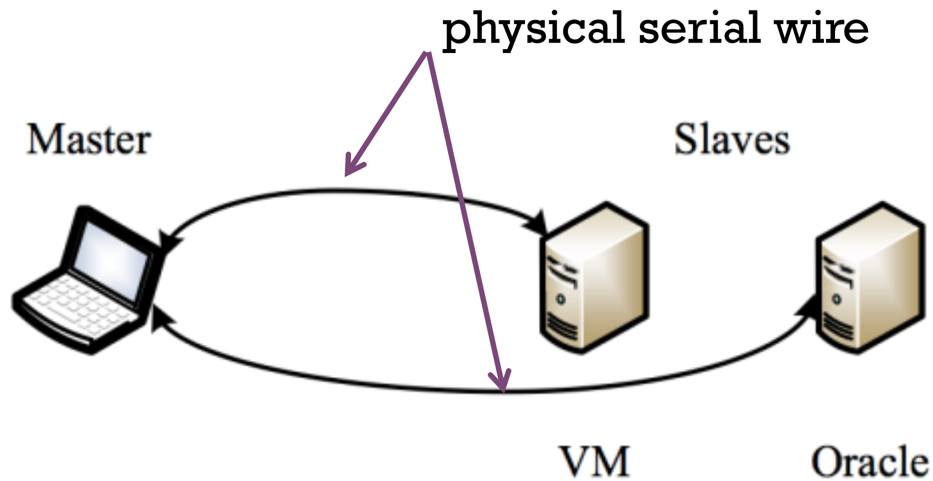
+ Overview

- Our Goal
  - Enumerate all the CPU semantic differences between a virtual machine (VM) and a physical machine (Oracle)
  - Lie to malware with the expected values for Oracle (like kernel rootkits)
- Results
  - We find 5 times more pills running 15 times fewer tests Red Pill testing
    - Almost half of our tests yield a pill
  - We analyze two root causes of pills
    - VM does not adhere to specs
    - Vague specs lead to different implementations in CPUs
  - Most pills stem from differences in kernel registers
    - even with hardware-assisted VMs
  - We can enumerate all differences between a VM and a physical machine for selected instructions

Related Work

- **Red Pill Testing** (EmuFuzzer, Martignoni09)
  - user-space)
  - Random tests cannot guarantee completeness
- **KEmuFuzzer** [Martignoni10] (extend to kernel-space)
  - ctions
  - Random tests cannot guarantee completeness
  - Custom kernel cannot be generalized
- **Hi-Fi tests for Lo-Fi emulators** [Martignoni12]
  - -fidelity emulator
  - Symbolic execution cannot test FPU instructions
  - code paths
  - Comparing two VMs is different from comparing a VM and a physical machine
  - lators

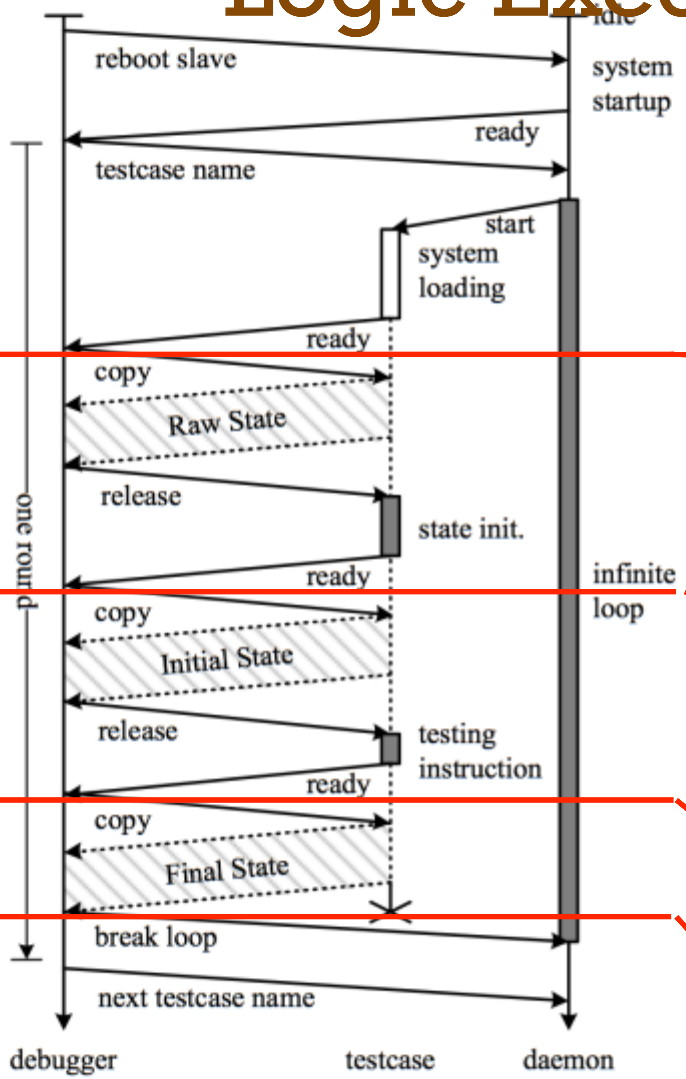
Cardinal Pill Testing
- Architecture Overview



- WinDbg 6.12
- Stores test case states in various testing phases
- Compare states to determine a pill

- Store test cases
- A daemon that helps communication between Master and a Slave
- **State**: all user and kernel registers, the data stored in the part of code, data, and stack segments that test cases read or write

Cardinal Pill Testing - Logic Execution



1. Master issues a test case name to the daemon in a Slave
   - The daemon loads the test case and then notifies Master
2. Master stores the Raw State and releases the Slave
   - The test case performs initialization work and notifies Master
3. Master stores the Init State and releases the Slave
   - The test case executes the testing instruction and notifies Master
4. Master stores and compares the Final State

+ Cardinal Pill Testing - Testing Goal

- Generate a minimal set of test cases for each instruction that explore all possible code paths
- Starting from Intel Manuals
  - **Defined Behaviors**: manuals have clear semantics for register modifications and exceptions
  - **Undefined Behaviors**: not specified by manuals
  - E.g., `aaa` adjusts the sum of two unpacked binary coded decimal to create an unpacked BCD result
    - Input: the `al` register and `AF` flag
    - Defined behavior: set `AF` and `CF` to `1` if there is a carry; otherwise `0`
    - Undefined behavior: `OF`, `SF`, `ZF`, and `PF` are undefined

Cardinal Pill Testing - Testing Goal

- For defined behaviors for a given instruction
  - Evaluate all code branches
  - Consider all flag bit states that are read implicitly or updated using results
- Evaluate all exceptions
  - E.g., memory access, invalid input arguments
- Investigate undefined behaviors
  - To reveal undocumented implementation specifics

Cardinal Pill Testing - Test Case Generation

- Instruction Grouping (Intel x86)
  - Classify instructions into five broad categories
    - Arithmetic, data movement, logic, flow control, and misc

Category	Instruction Count	Example Instructions	Parameter Coverage
arithmetic	48	`aaa, add, imul, shl, sub`	min, max, boundary values, randoms in different ranges
	336	`addpd, vminss, fmul, fsqrt, roundpd`	±infi, ±normal, ±denormal, ±0, SNaN, QNaN, QNaN floating-point indefinite, randoms
data mov	232	`cmova, fild, in, pushad, vmaskmovps`	valid/invalid address, condition flags, different input ranges
logic	64	`and, bound, cmp, test, xor`	min, max, boundary values, >, =, <, flag bits
	128	`andpd, vcomiss, pmaxsb, por, xorps`	±infi, ±normal, ±denormal, ±0, SNaN, QNaN, QNaN FP indefinite, >, =, <, flag bits
flow ctrl	64	`call, enter, jbe, loopne, rep stos`	valid/invalid destination, condition flags, privileges
misc	34	`clflush, cpuid, mwait, pause, ud2`	analyze manually and devise dedicated input

+ Cardinal Pill Testing - Test Case Generation

- Arithmetic Group
  - Instructions first read arguments and then perform arithmetic operations
  - Combine instructions that read/write the same registers with similar rules into a partition
  - E.g. `aaa`, `aas`, `daa`, and `das`
    - Compare the `al` register with `0fh` and check the adjustment flag `AF`
  - Test cases for this partition
    - Initialize `al` to min (`00h`), max (`0ffh`), boundary (`0fh`), random values in different ranges ([`01h`, `0eh`], [`10h`, `0feh`])
    - Also flip `AF` between clear and set for different `al` values

Evaluation - Overview

- Test case generated from Intel IA-32 manual
  - 1,653 instructions, counting different addressing modes
  - 906 unique mnemonics
  - ~ 230 groups, ~ 1.5 human months to generate test cases
  - 19,412 test cases in total
- Infrastructure and Software
  - QEMU (4 versions)
    - Tiny Code Generation mode: pure-software translation
    - VT-x (Intel hardware assisted, high fidelity)
  - Bochs 2.6.2 (pure-software translation)
  - **Oracle 1**: Intel Xeon E3 3.40GHz, Windows7 Pro x86
  - **Oracle 2**: Intel Xeon W3520 2.6GHz, Windows XP x86 SP3

+ Evaluation - Overview

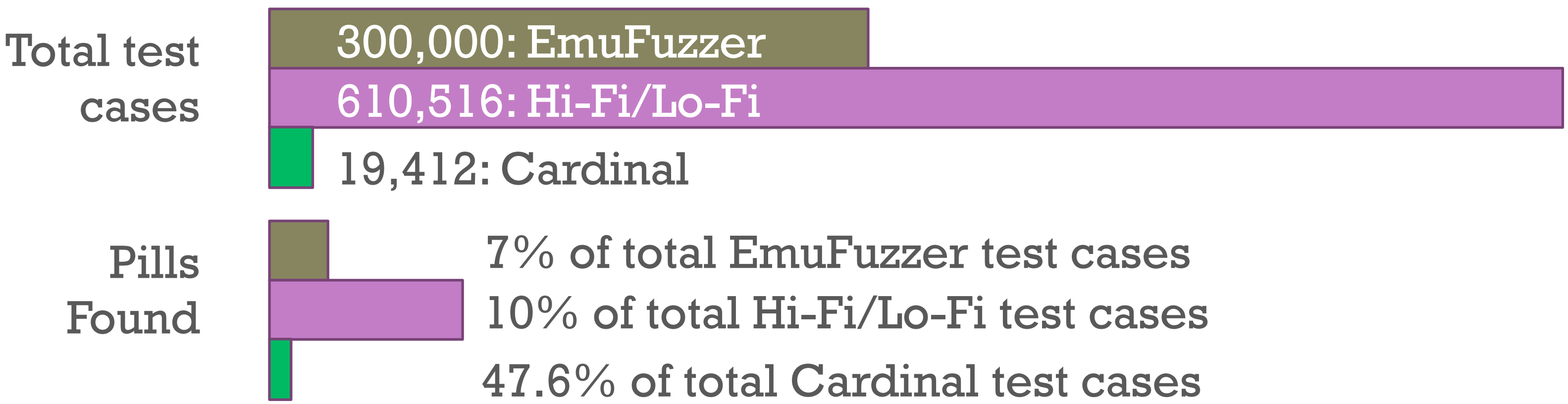
- Results (# test cases out of a total 19,412)

VMs	#Pill / Pct.	#Crashed / Pct.	#Fatal / Pct.
Q1 (TCG)	9,255 / 47.7%	7 / < 0.1%	1,378 / 7%
Q2 (TCG)	9,201 / 47.4%	7 / < 0.1%	1,376 / 7%
Q1 (VT-x)	7,523 / 38.7%	2 / < 0.1%	3 / < 0.1%
Q2 (VT-x)	7,478 / 38.5%	2 / < 0.1%	0
Bochs	8,958 / 46.1%	2 / < 0.1%	950 / 4.9%

- **Crashed test cases**: crash itself or the system
- **Fatal test cases**: VM and Oracle exhibit different initial states
  - Due to VM implementation bugs
- **Note**: Although Intel VT-x provides high-fidelity, the actual performance depends on how VMs utilize it

+ Evaluation - Comparison with Related Work

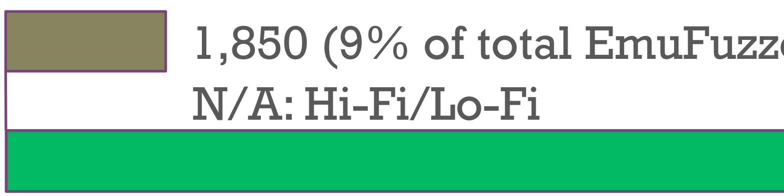
- High yield = faster testing



  - We achieve a much larger yield rate (47.6%)!

Evaluation
- Comparison with Related Work

- All our pills are unique = faster testing
  - Unique Pills – different parameter values read by an instruction
    - E.g. the same pill:
      - **`Mov ebx, 80h; mov al, 9h; aaa;`**
      - **`Mov ebx, 0ffh; mov al, 9h; aaa;`**
      - Because **`aaa`** does not read **`ebx`** at all!



1,850 (9% of total EmuFuzzer pills)

N/A: Hi-Fi/Lo-Fi

9,255 (100% of total Cardinal pills)

+ Evaluation
- Comparison with Related Work

- We cover all mnemonics = guarantee completeness
  - Unique Mnemonic
    - E.g., aaa, aad, fmul

  - Our pills cover 4 times more mnemonics than those found by EmuFuzzer!

+ Evaluation - Root Causes of Pills (Defined)

- Root causes listed in Hi-Fi/Lo-Fi work
  - We find:
    - Pills in general purpose instructions
    - Pills due to QEMU's memory management unit
  - We do not find:
    - Pills in kernel instructions
      - E.g., `iret` pops items from stack differently from a physical machine
    - Because we do not extensively test kernel instructions
      - Due to extensive time required for testing kernel instructions
      - We leave this for future work

+ Evaluation
- Root Causes of Pills (Defined)

- Our new findings about QEMU
  - Incorrect 6 flags and 8 masks in `mxcsr` register when no exception happens
    - invalid operation, denormal flag, precision mask.
  - Incorrect 7 flags in `fpsw` status register
    - stack fault, FPU busy.
  - Fails to throw 5 exceptions
    - float_multiple_traps, float_multiple_faults, etc
  - Incorrect `fptw` tag register
    - sets to "zero" when it should be "empty", etc
  - Incorrect floating-point instruction pointer and data pointer
  - Please check our paper and website for the detailed list.
    - http://steel.isi.edu/Projects/cardinal

+ Evaluation - Causes of Pills (Undefined)

- The only source is EFLAGS register
  - Generate additional test cases to explore the semantics of modifications to undefined resources in each CPU
  - A flag may be (1) cleared, (2) intact, (3) set according to ALU output at the end of an instruction's execution, or (4) set based on an ALU output of an intermediate operation
  - We devise a testing method to differentiate between these cases (more details in the paper)
- Understand the semantics of undefined resource modification
  - Help devise hiding rules without exhaustive tests

Summary

- We propose Cardinal Pill Testing:
  - Moderate manual effort to analyze instructions in a manual, then automated test generation and pill identification
  - Our tests have high yield and superb coverage, compared to related work
- Completeness?
  - Pills for user-space instructions that affect defined resources (stem from incorrect VM implementation) - complete
  - Pills that affect undefined resources (stem from different implementations in physical machines) – complete only for a given VM/physical machine pair
  - We did not extensively test pills that relate to kernel-space instructions due to high test-time demand – incomplete
- Propose a way to lie to malware via modification of VM translation engine
  - Details in the paper

Questions?

- Thank you!!
- Contact us:

  - Hao Shi haoshi@usc.edu
  - Abdulla Alwabel alwabel@isi.edu
  - Jelena Mirkovic mirkovic@isi.edu

  - http://steel.isi.edu/Projects/cardinal