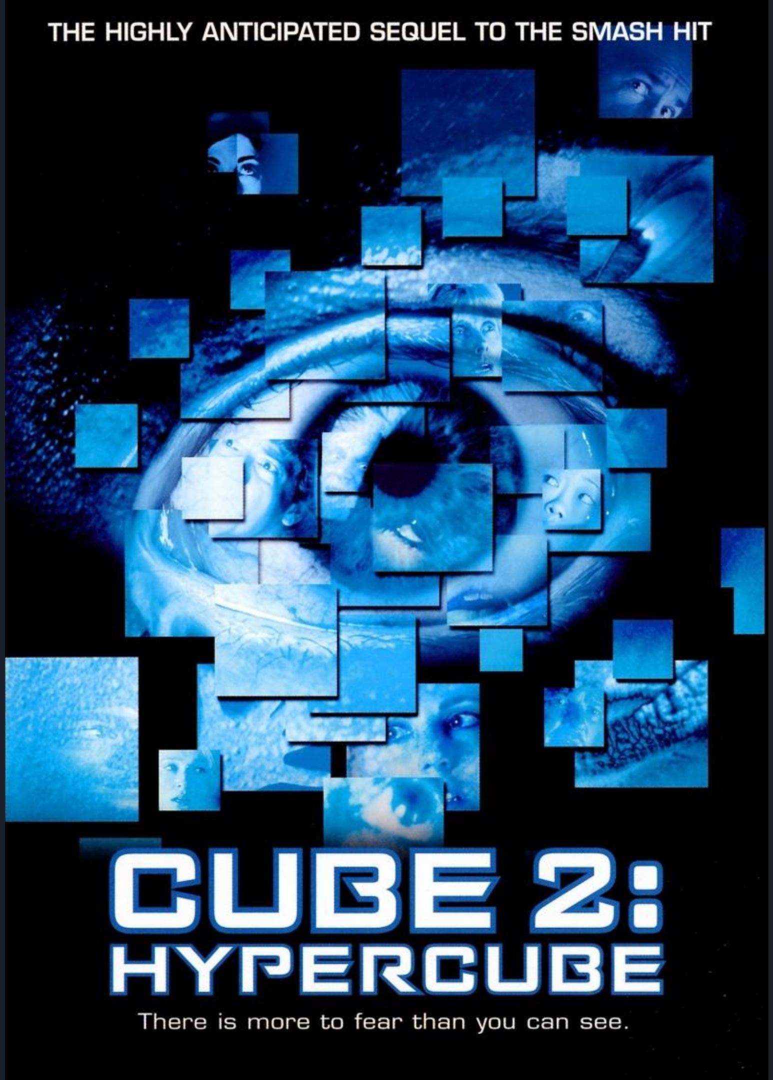
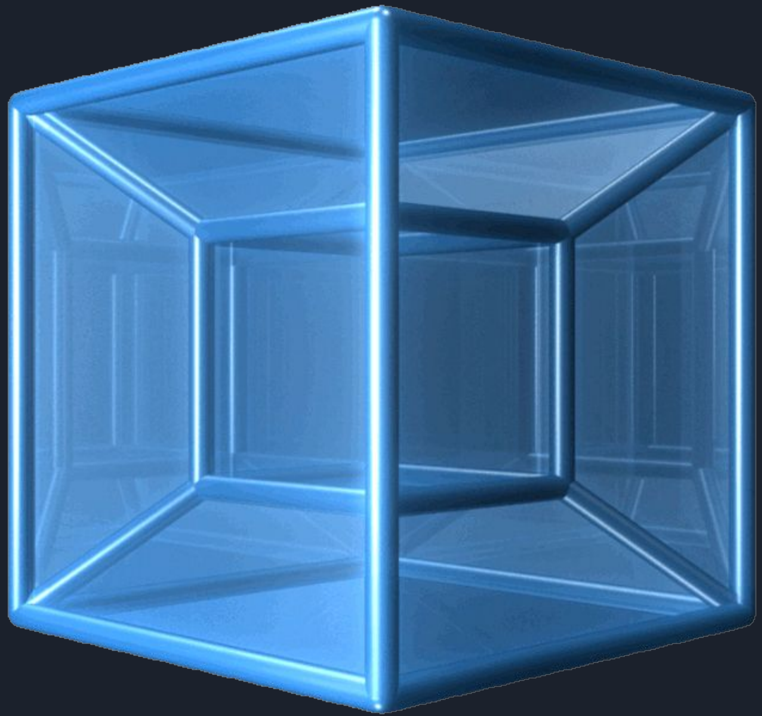
HYPER-CUBE

HIGH-Dimensional Hypervisor Fuzzing


THE HIGHLY ANTICIPATED SEQUEL TO THE SMASH HIT
CUBE 2:
HYPERCUBE
There is more to fear than you can see.



HYPER-CUBE

At a high-level HYPER-CUBE is a blackbox fuzzer designed to test hypervisors

HYPER-CUBE components

- HYPER-CUBE OS - small, specialized OS
- TESSERACT - byte code interpreter for fuzzing
- A few accessory tools for working with TESSERACT

Claimed contributions

1. Multi-dimensional, platform-independent fuzzing method
2. Fast hypervisor fuzzing
3. Custom OS for hypervisor fuzzing

Prior Work

- VDF - AFL fuzzer for QEMU device emulators
- IOFUZZ - random writes to port mapped I/O
- Intel CHIPSEC - security tool that can fuzz hardware interfaces

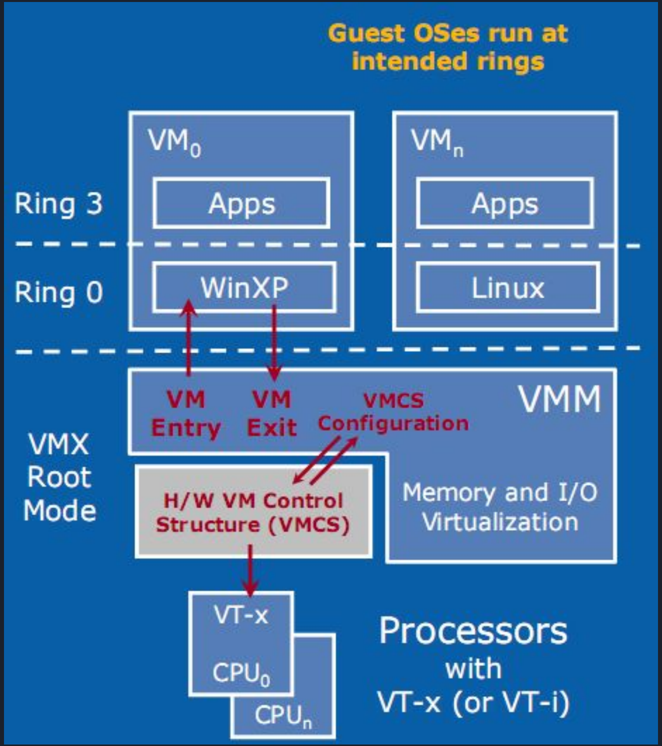
X86 Virtualization Extensions

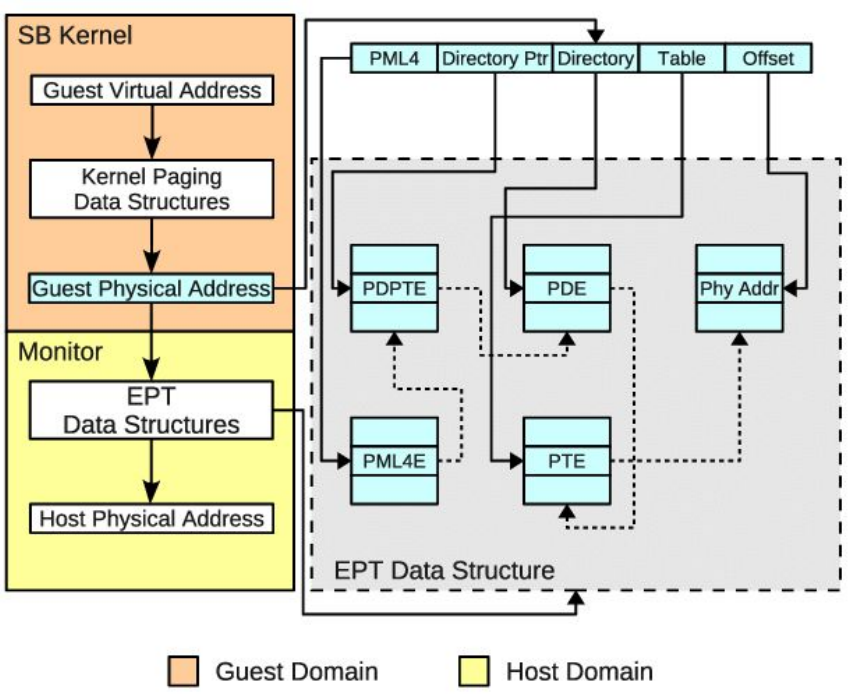
There extensions to X86 that add hardware acceleration to virtualization, Intel VT-x and AMD-V.

A few key extensions

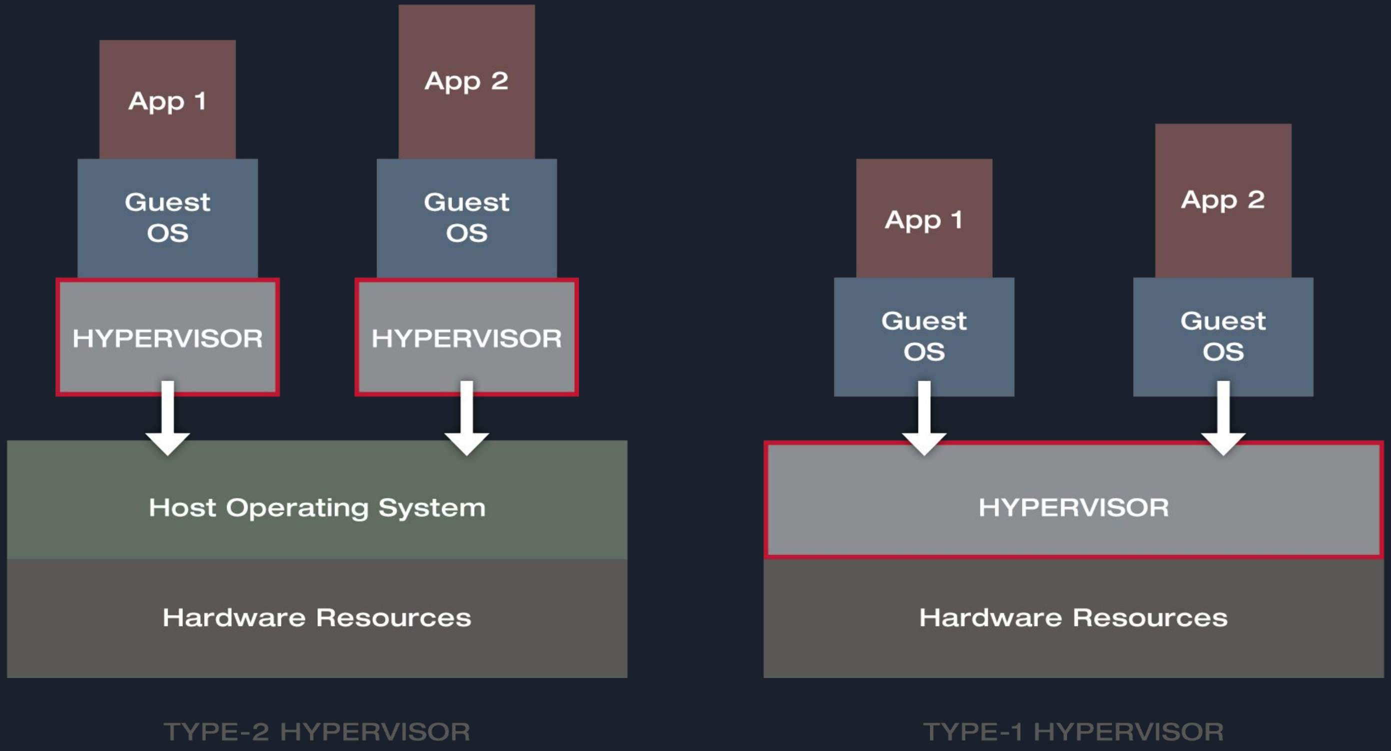
- VMX - adds 13 new instructions, two new CPU modes
  - Root mode, used by the hypervisor
    - Can configure the Virtual Machine Control Structure (VMCS)
  - Non-root mode, used by guests
- Extended Page Tables (EPT)
  - Adds another level of page table translation
- Interrupt Virtualization
- I/O MMU Virtualization

X86 Virtualization Extensions

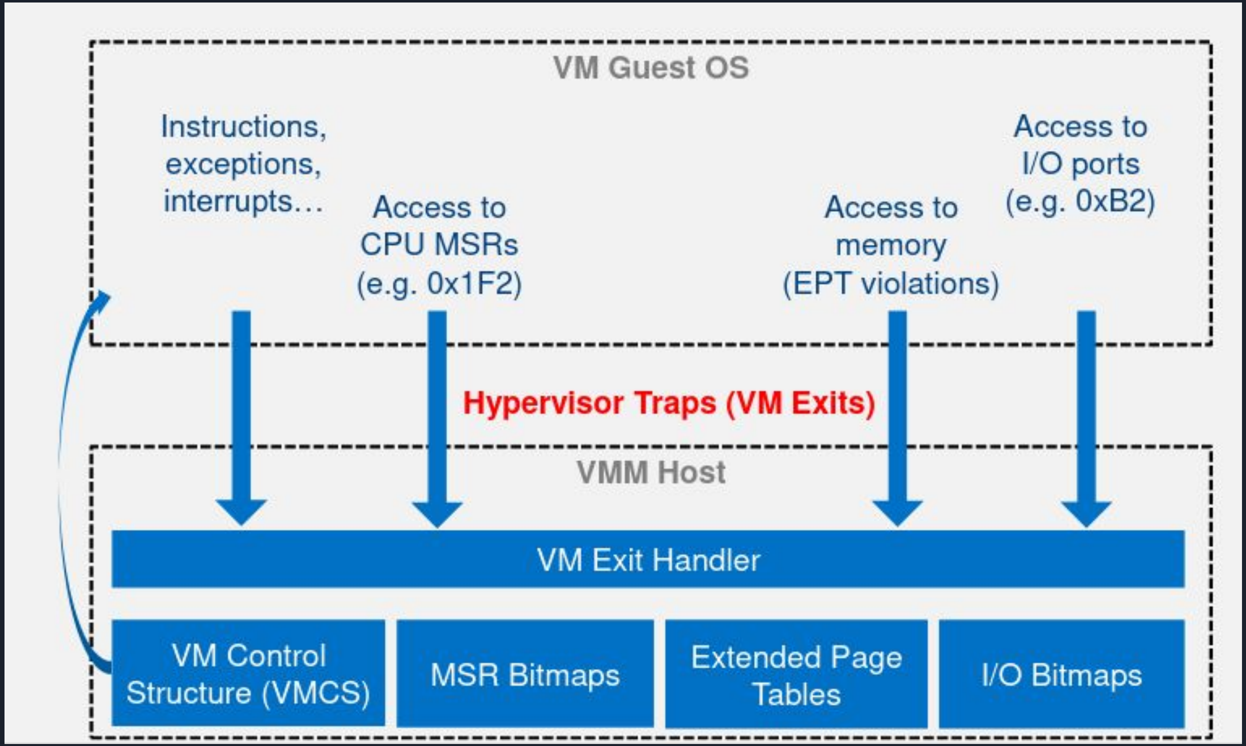




Hypervisor Background



VMExit


VM Guest OS
Instructions, exceptions, interrupts…
Access to CPU MSRs (e.g. 0x1F2)
Access to memory (EPT violations)
Access to I/O ports (e.g. 0xB2)
Hypervisor Traps (VM Exits)
VMM Host
VM Exit Handler
VM Control Structure (VMCS)
MSR Bitmaps
Extended Page Tables
I/O Bitmaps

VMExit

- Unconditional exit
  - VMX/SVM instructions
  - CPUID
  - GETSEC
  - INVD
  - XSETBV
- Conditional exit
  - CLTS
  - HLT
  - IN, INS/INSB/INSW/INSD, OUT, OUTS/OUTSB/OUTSW/OUTSD
  - INVLPG
  - INVPCID
  - LGDT, LIDT, LLDT, LTR, SGDT, SIDT, SLDT, STR
  - LMSW
  - MONITOR/MWAIT
  - MOV from CR3, CR8 / MOV to CR0, CR3, CR4, CR8
  - MOV DR
  - PAUSE
  - RDMSR/WRMSR
  - RDPMC
  - RDRAND
  - RDTSCP
  - RSM
  - WBINVD
  - XRSTORS / XSAVES

Design Goals

- X86 Hypervisor agnostic
- Blackbox fuzzing with high throughput
- High-dimensional
  - Interfaces
  - Operations

OS/Fuzzing Architecture

HYPER-CUBE OS

1. Multiboot 2 spec compliant, uses GRUB for booting OS
   a. Can be boot via BIOS or UEFI
2. Simply memory management system
   a. Uses a heap design and virtual addressing. Some virtual addresses map directly to physical address.
3. Interface enumerator
   a. Finds most attached hardware interfaces
4. TESSERACT byte-code interpreter for fuzzing
5. Serial out interface
   a. For outputting results

Interfaces and enumeration

- Memory-mapped I/O (MMIO)
  - High precision event timers
  - Advanced programmable interrupt controller
  - PCI/PCIe devices
- Legacy Port I/O (PIO)
  - Programmable interrupt controller
  - ISA devices
  - Not exhaustive
- Direct Memory Access (DMA)
  - Not enumerated
- Hypercalls
  - Hardcoded, specific to each hypervisor

TESSERACT

TESSERACT is a bytecode interpreter that generates calls to hypervisor interfaces.

- Is given a seed or generates ones. Seed is uses to feed pseudorandom generator for creating a stream
- MMIO operations are given a region_id and offset. Is modulus size of respective range.
- Can be given dictionaries of data values and offsets

TESSERACT

- write_mmio(region id, offset, data) writes a single word data to the address given by region_id+offset.
- read_mmio(region id, offset) reads a single word from the address given by region_id+offset.
- xor_mmio(region id, offset, mask) reads a single word from the given address, and writes it back after applying the given XOR mask.
- bruteforce_mmio(region id, offset, data, num) writes num consecutive data words to the given address.
- memset_mmio(region id, offset, data, num) writes the word data to num consecutive addresses, beginning at the given address.

TESSERACT

- writes_mmio(region id, offset, data, num) same as memset mmio, however it uses a rep prefixed instruction to perform the task, testing instruction emulation.
- reads_mmio(region id, offset, num) same as writes mmio, but instead of writing data, it reads it.
- mmio_write_scratch_ptr(region id, offset, scratch-id, scratch-offset) writes a pointer to the given offset in the scratch area to the address in the given MMIO region.
- *_io() all opcodes accessing MMIO regions are implemented for I/O ports as well.
- write_msr(msr num, mask) writes to a MSR. This operation is limited to a list of ≈ 240 well-known MSRs. The mask is xored into the selected MSR.
- hypercall(eax, ebx, ecx, edx, esi) executes arbitrary hypercalls using the given registers as arguments. KVM
- vmport(ecx, ebx) executes arbitrary vmport hyper-calls with the registers set to the arguments to the hypervisor. VMware

TESSERACT

- writes_mmio(region id, offset, data, num) same as memset mmio, however it uses a rep prefixed instruction to perform the task, testing instruction emulation.
- reads_mmio(region id, offset, num) same as writes mmio, but instead of writing data, it reads it.
- mmio_write_scratch_ptr(region id, offset, scratch-id, scratch-offset) writes a pointer to the given offset in the scratch area to the address in the given MMIO region.
- *_io() all opcodes accessing MMIO regions are implemented for I/O ports as well.
- write_msr(msr num, mask) writes to a MSR. This operation is limited to a list of ≈ 240 well-known MSRs. The mask is xored into the selected MSR.
- hypercall(eax, ebx, ecx, edx, esi) executes arbitrary hypercalls using the given registers as arguments. KVM
- vmport(ecx, ebx) executes arbitrary vmport hyper-calls with the registers set to the arguments to the hypervisor. VMware

PRNG Stream

```
                ...
0120: 2fff 1c27 ab47 5700
0128: adf2 3d60 092f 5488
0130: ec2d 9d1a 029d 56fd
0138: e0d1 a275 1f56 1d28
0140: ea78 a2fa db07 d60d
0148: 1288 3a5a 91f9 1756
0150: 1cae 31ad 9b9c 938e
0158: 2a33 f597 6615 e267
0160: 0117 1f16 b440 8a86
0168: 9154 5b55 e4ca 9e3d
0170: 9d19 ae79 efac e500
0178: 8cdf 8c00 9a83 df76
0180: 91fe d779 026c 2e2b
0188: 9137 1ef8 eea3 d29c
0190: 1789 5938 a36f 718a
0198: 81e4 678c 20f5 fa0b
01a0: 774d 07f1 cee3 62bc
01a8: d845 bc86 7631 6eac
                ...
```

Robust Interpretation

Opcode Handler

vmport(0xbd4,0x10ea)
memset_io(0x426,0xce0,0x9dc,0xca8)

writes_mmio(0xec8,0xad,0x10ac,0x7e9)

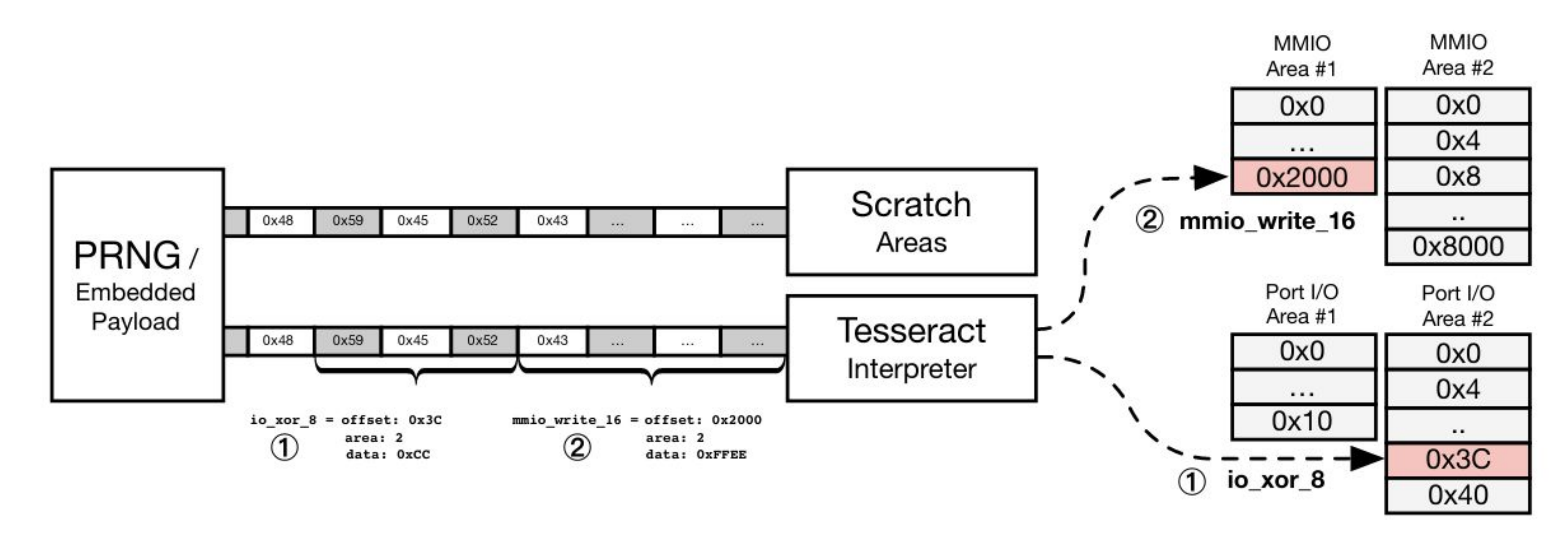
bruteforce_mmio(0xce4,0xdfa,0xe31,0x322)

writes_io(0x4bb,0xb8,0xeb1,0x401)

memset_mmio(0x128,0xa73,0x2b3,0xa84)
read_mmio(0xbf3,0x907)

bruteforce_io(0x5c4,0x49a,0x94f,0xb1c)

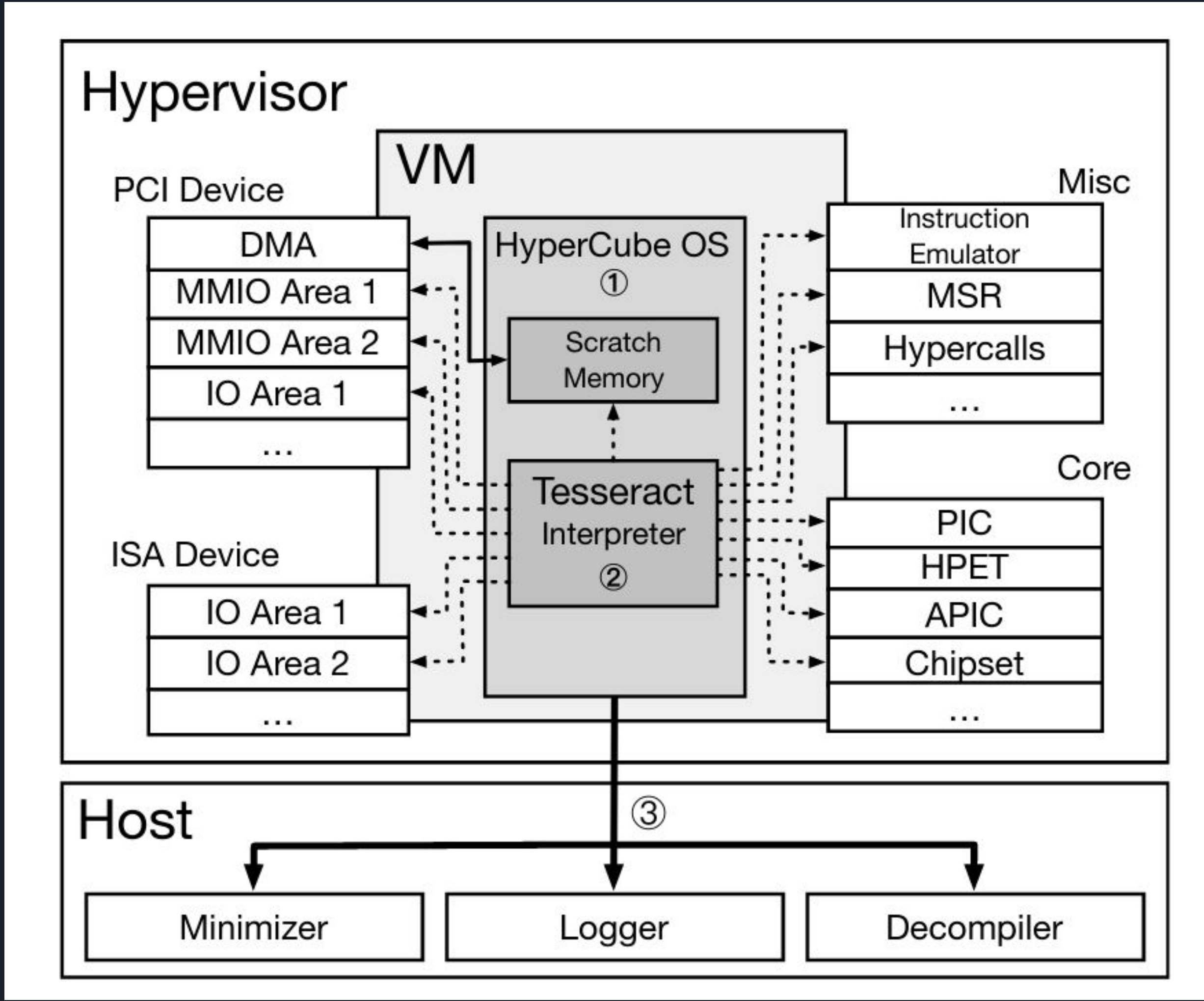
xor_mmio(0x54b,0xa00,0xb51)



PRNG / Embedded Payload
0x48
0x59
0x45
0x52
0x43
...
...
...
Scratch Areas
Tesseract Interpreter
io_xor_8 = offset: 0x3C
area: 2
data: 0xCC
①
mmio_write_16 = offset: 0x2000
area: 2
data: 0xFFEE
②
MMIO Area #1
0x0
...
0x2000
MMIO Area #2
0x0
0x4
0x8
..
0x8000
② mmio_write_16
Port I/O Area #1
0x0
...
0x10
Port I/O Area #2
0x0
0x4
..
0x3C
0x40
① io_xor_8

External Tools

- Logger
  - Accepts serial communication from virtual machine
- Results minimizer
  - Tries to reduce triggering input stream down to smallest possible trigger
- Decompiler
  - For interrupting results


Hypervisor
VM
PCI Device
DMA
MMIO Area 1
MMIO Area 2
IO Area 1
...
ISA Device
IO Area 1
IO Area 2
...
HyperCube OS
①
Scratch Memory
Tesseract Interpreter
②
Misc
Instruction Emulator
MSR
Hypercalls
...
Core
PIC
HPET
APIC
Chipset
...
Host
③
Minimizer
Logger
Decompiler

Tested Hypervisors

- FreeBSD bhyve (12.0-RELEASE)
- VirtualBox (5.1.37_Ubuntu r122592)
- Parallels Desktop (14.1.3)
- KVM/QEMU (4.0.1-rc4)
- Intel ACRN (29360 Build)
- VMware Fusion (11.0.3)

Results

Found 55 bugs, 43 CVEs

- Assert Failures 25
- Null-Pointer Dereferences 13
- Memory-Corruptions 8
- Div-By-Zero (FP Exceptions) 5
- Deadlocks 4

HYPER-CUBE vs VDF

Fuzzed 15 Device Emulators

HYPER-CUBE

- 13/15 More Coverage
- 9/15 Crashes
- 10 Minutes Each

VDF

- 2/15 More coverage
- 4/15 Crashes
- ~ 60 days each

Limitations

1. Restrictions on some Type-1 hypervisors
2. Paravirtualized components need hand written interfaces
3. Black-box fuzzing