Zhejiang University

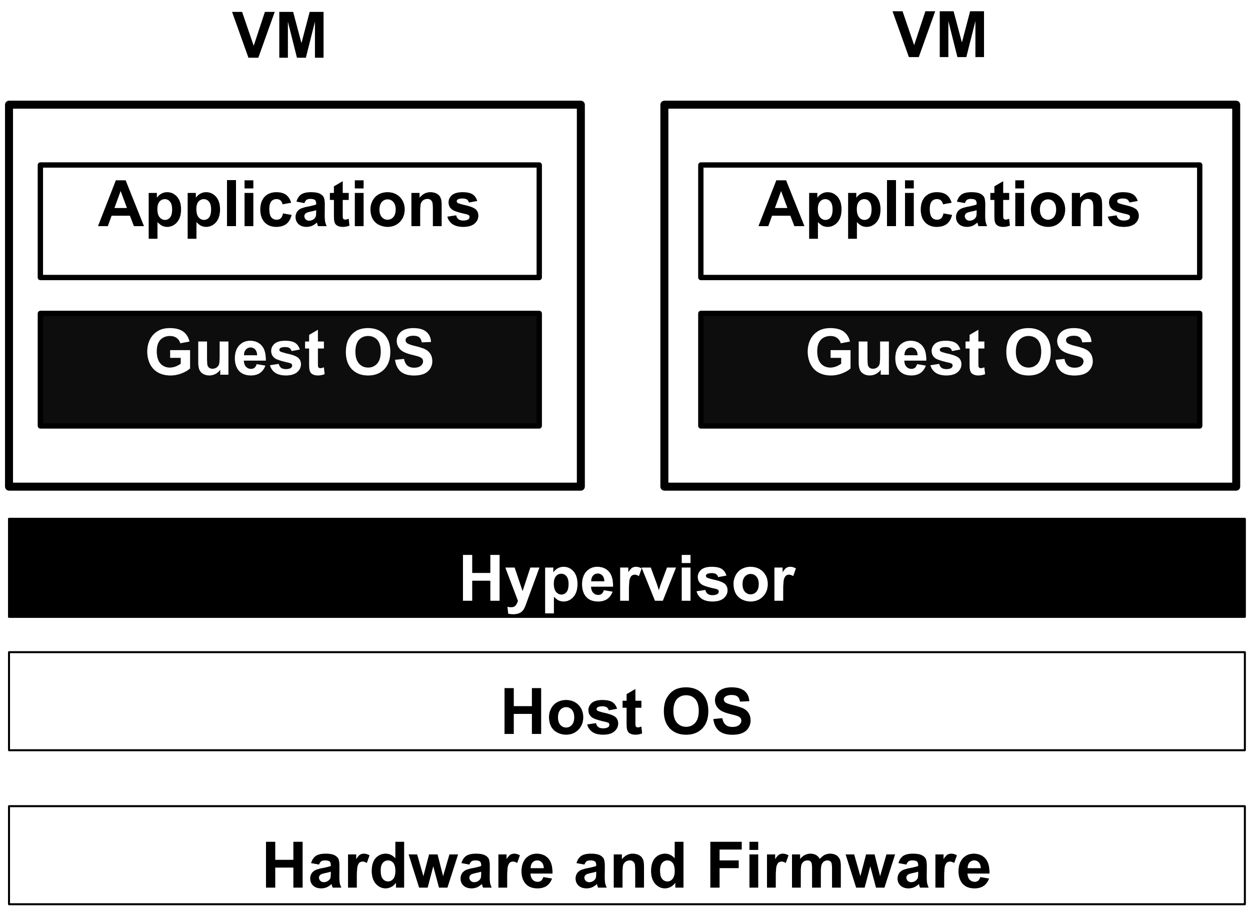
ANT GROUP | Alipay

V-SHUTTLE: Scalable and Semantics-Aware Hypervisor Virtual Device Fuzzing

Gaoning Pan   Xingwei Lin   Xuhong Zhang   Yongkang Jia
Shouling Ji   Chunming Wu   Xinlei Ying   Jiashui Wang   Yanjun Wu

CCS 2021

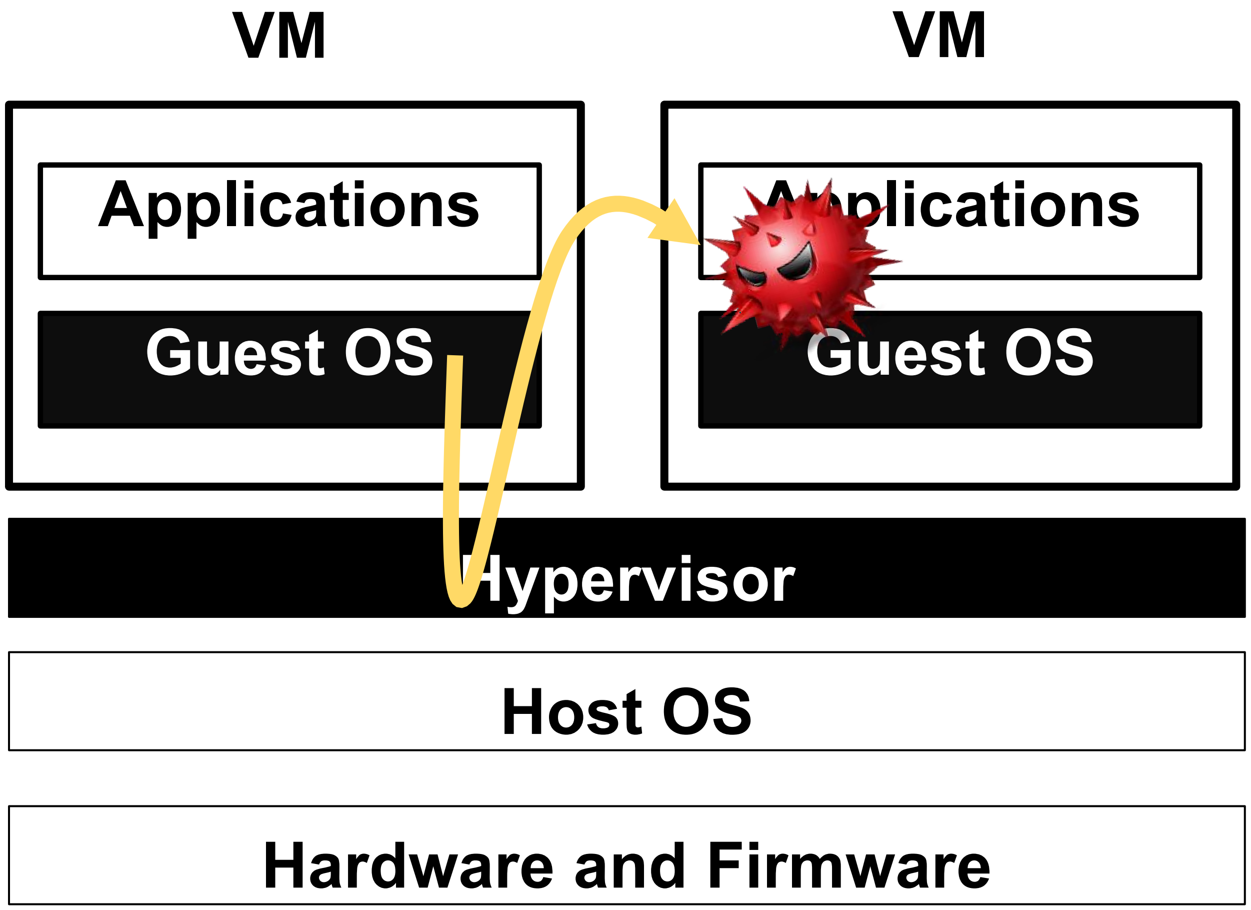
Background



Virtual Machine Architecture

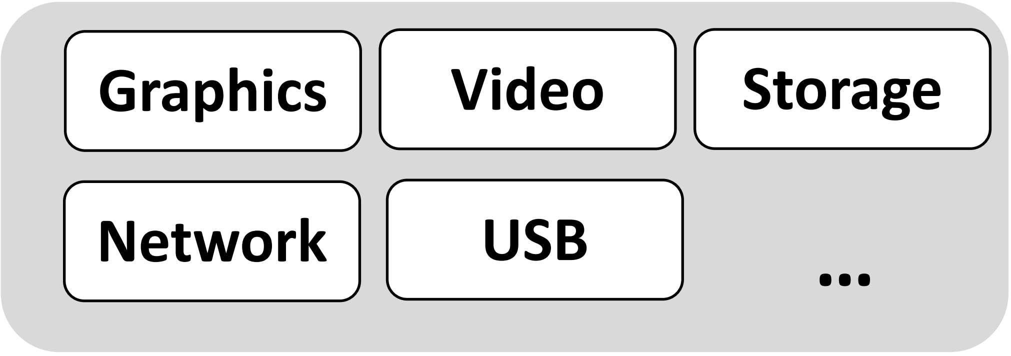
- **Virtual machine architecture**
  - uses hypervisor, a.k.a. VMM
  - provides strong isolation with virtualized hardware
  - has each execution environment

Background



Virtual Machine Architecture

- **Virtual Machine Escape**
  - A hypervisor has lots of code for services and hardware emulations
  - Privilege escalation: Guest -> Host



Background

Existing VM escape

- Storage device: Scavenger [Blackhat Asia’ 21], VENOM
- Graphics device: 3d Red Pill [Blackhat Asia’ 20]
- ……

- High bounty target in famous PWN competitions, like Pwn2Own and TianfuCup

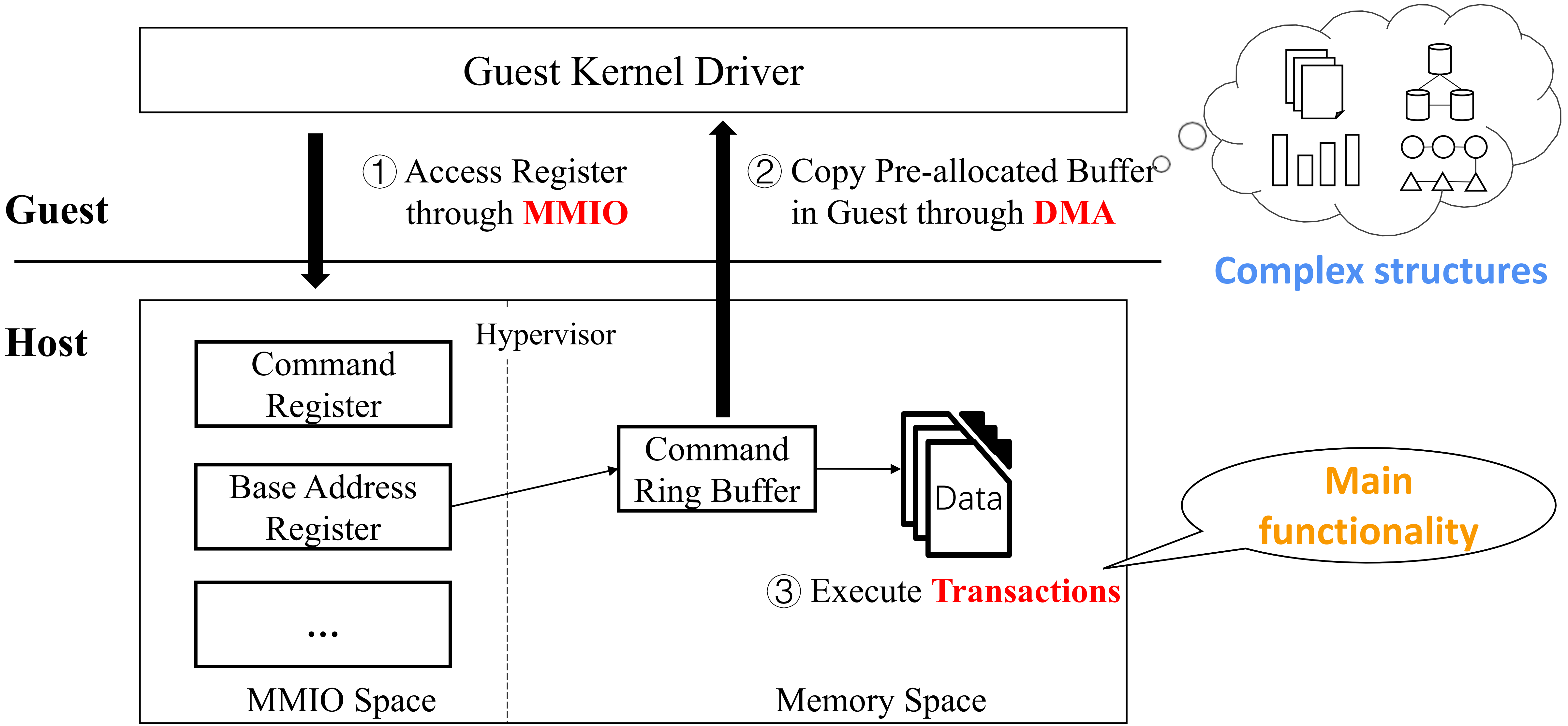
TianfuCup @TianfuCup · Nov 7, 2020

The escape from #qemu is confirmed! Two bugs exploited: a uaf and an information-disclosure bug. $60,000 awarded to 360 ESG Vulnerability Research Institute @XiaoWei___ Congrats!

> **TianfuCup** @TianfuCup · Nov 7, 2020
>
> Wow, @XiaoWei___ contributed another successful entry, it's against target Ubuntu + qemu-kvm. VM escape achieved. Excelleeeeent!

2 14 46

Virtual Device Transaction



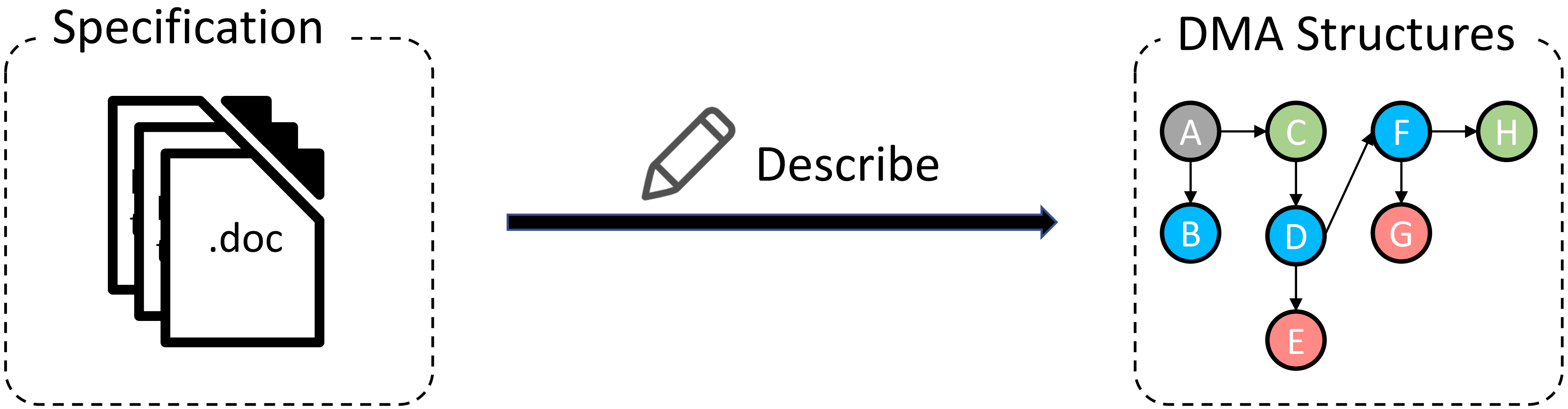
Study on DMA

- 5 most popular QEMU device categories used in virtualization scenarios

Category	Device (support DMA)	Number	Total
USB	uhci, ehci, ohci, xhci	4	4
Storage	esp, ahci, lsi53c810, megasas, mptsas, nvme, pvscsi, sdhci, virtio-blk, virtio-scsi, virtio-9p	11	12
Network	e1000, e1000e, eepro100, pcnet, rocker, rtl8139, tulip, vmxnet3, virtio-net	9	10
Display	(null)	0	7
Audio	ac97, cs4231a, es1370, intel-hda, sb16	5	7
Avg		**29(72.5%)**	40(100%)

72.5% of the devices support DMA and use it to transfer complex data

Core Challenge – Nested Structures



Feature1: Nested Form Construction

- ➢ **Overall Level**: Higher-level tree structures and recursively defined.
- ➢ **Node Level**: Unknown pointer offset and unknown following node's address.

Feature2: Node Type Awareness

- ➢ **Overall Level**: Precise pointing relationships can only be known at runtime.
- ➢ **Node Level**: Fine-grained semantics of referred node types.

Motivating Example – USB UHCI

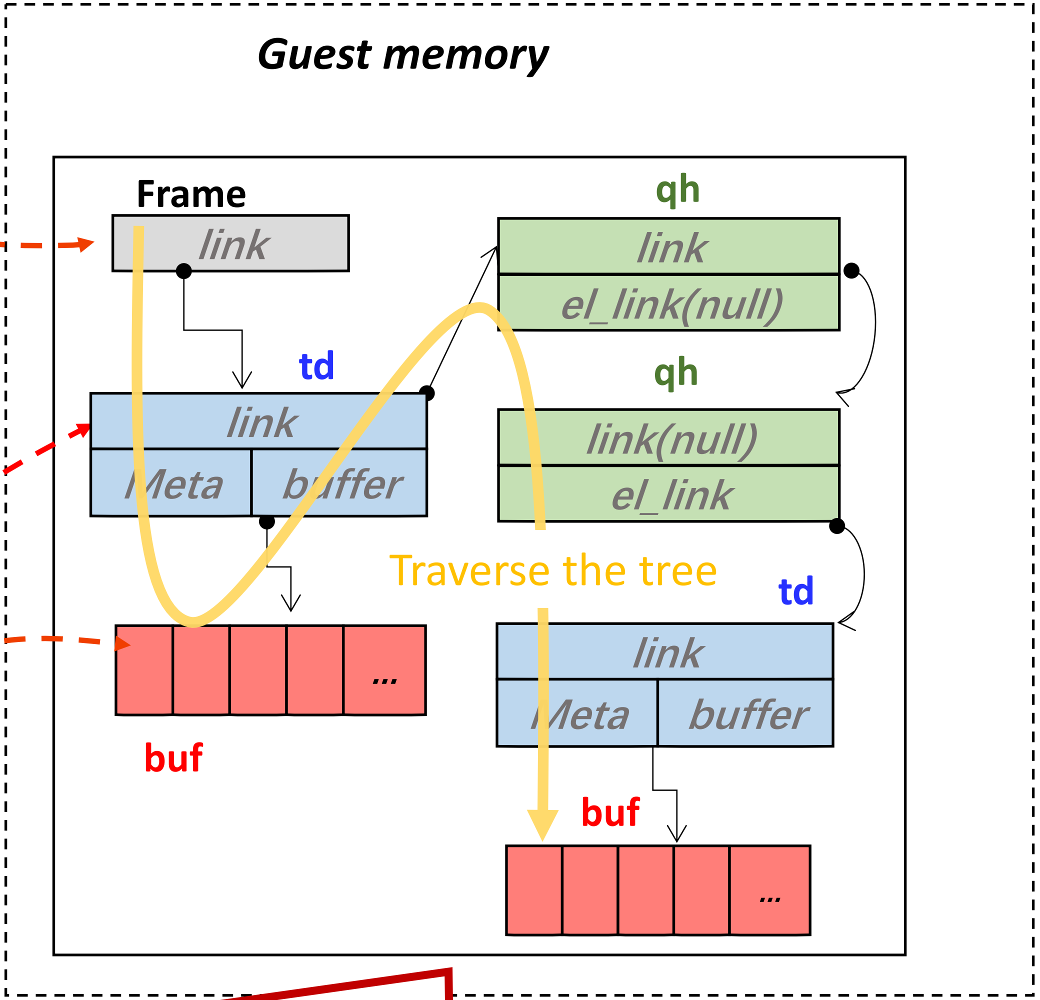
QEMU source code

dst addr

src addr

recursively

```
pci_dma_read (&s->dev, s->frame_addr, &link, 4);
…
for (; is_valid (link);) {
    …
    if (is_qh (link)) {
        pci_dma_read (&s->dev, link, &qh, sizeof(qh));
        …
        continue;
    }
    pci_dma_read (&s->dev, link, &td, sizeof(td));
    …
    pci_dma_read (&s->dev, td.buffer, buf, td.len);
    …
    /*   main usb packet processing   */
    link = td.link;
}
```



Q. How can such hierarchically nested structures be generated exactly?

Related Work

- Random fuzzing to basic interfaces (MMIO, DMA, etc.):
  - VDF [Andrew et al., RAID'17]
  - Hyper-Cube [Schumilo et al., NDSS'20]

No knowledge of the protocol implementation about DMA structures

- Apply expert-defined specifications to bridge the gap
  - Build structure-aware fuzzing against specifications that describe structures
  - Nyx [Schumilo et al., Security'21]

Structure-specific rules heavily rely on domain knowledge (**time-consuming)**

Related Work

- Random fuzzing to basic interfaces (MMIO, DMA, etc.):
    - VDF [Andrew et al., RAID'17]

Can we avoid such complex data structures building issues and make the fuzzing process fully automatic as well as domain knowledge free?

- Build structure-aware fuzzing against specifications that describe structures
- Nyx [Schumilo et al., Security'21]

Structure-specific rules heavily rely on domain knowledge (time-consuming)

Key Insight

Nested DMA Structures

Decoupled DMA Structures

Framework

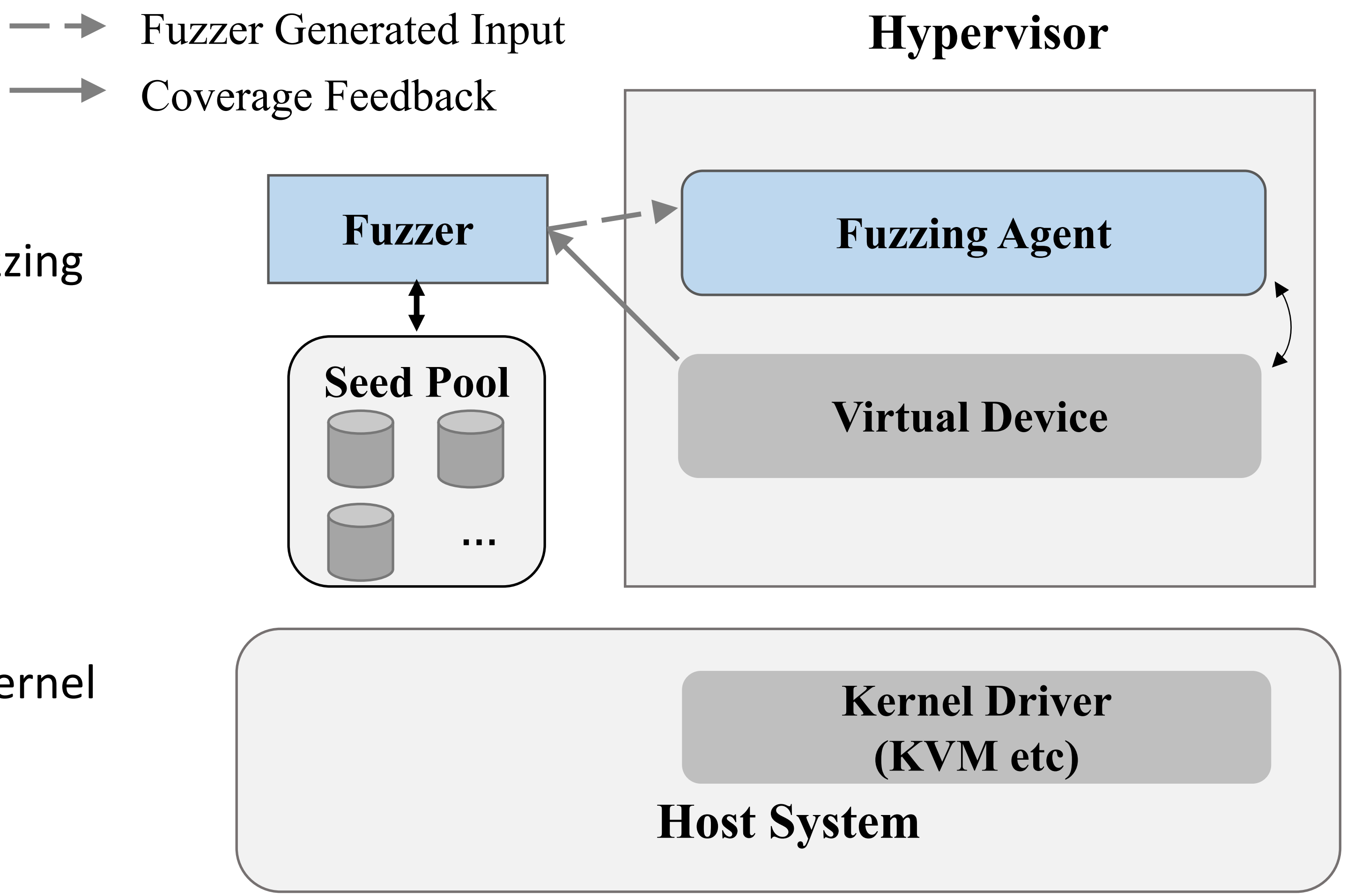
Overview of V-SHUTTLE

➢ Fuzzer

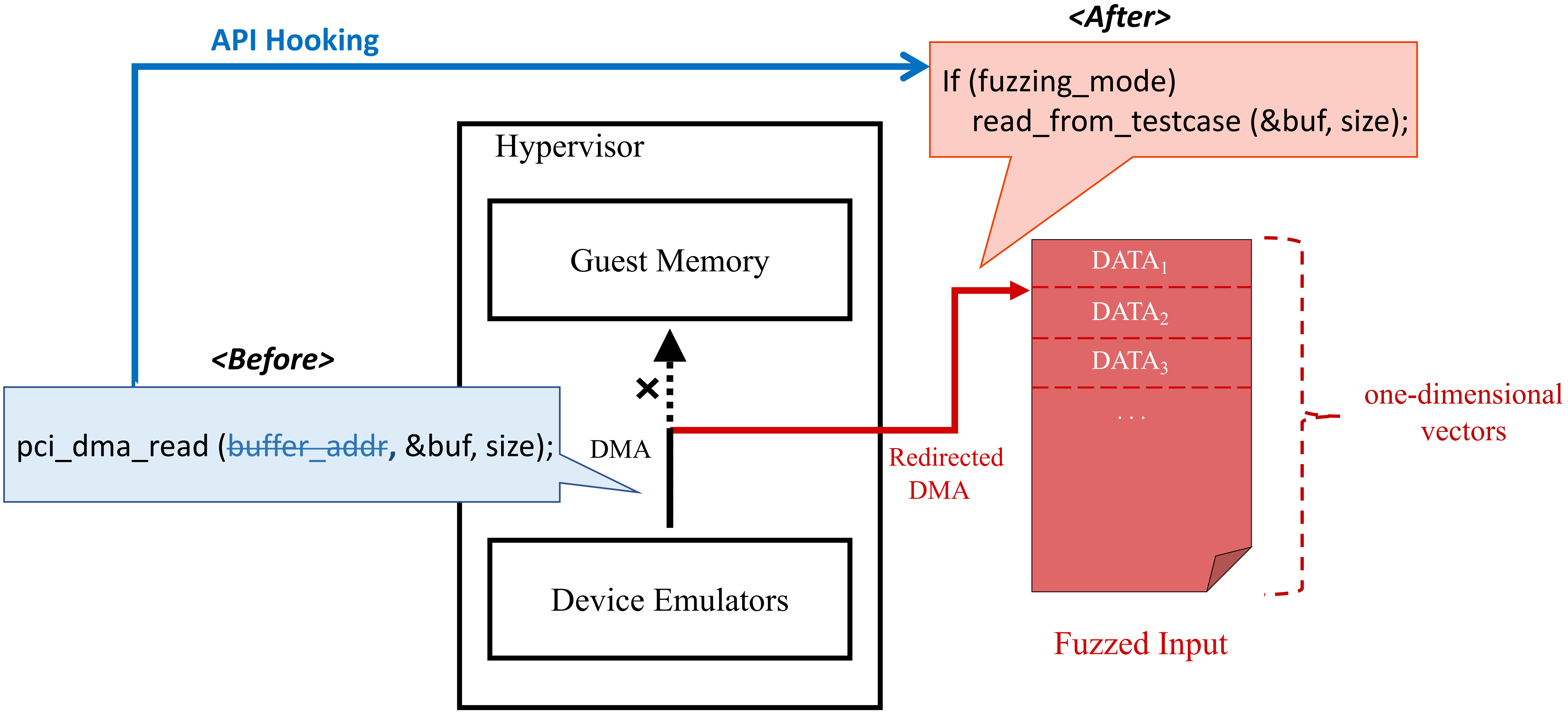
- Runs in host system
- Persistent mode to enable long-term fuzzing
- Collect coverage feedback
- Semantics-aware fuzzing via seedpools

➢ Fuzzing Agent

- Runs in the hypervisor
- Emulate malicious drivers of the guest kernel
- Intercept all DMA and I/O accesses

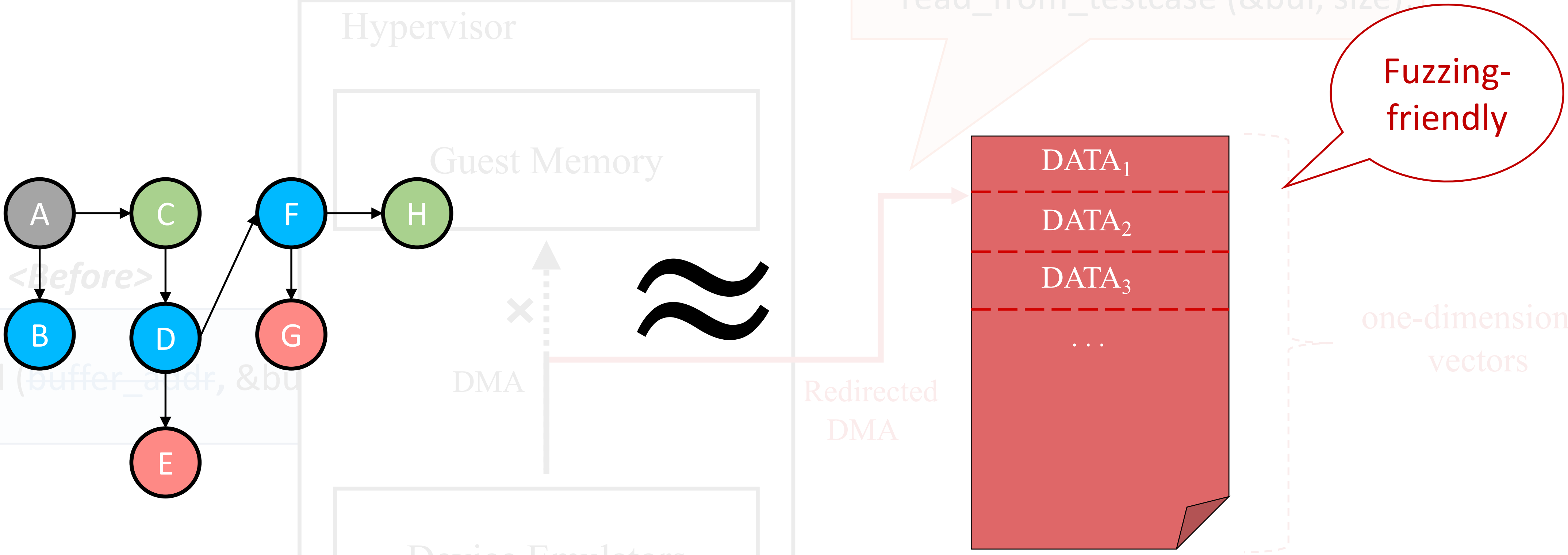


1. DMA Redirection



1. DMA Redirection

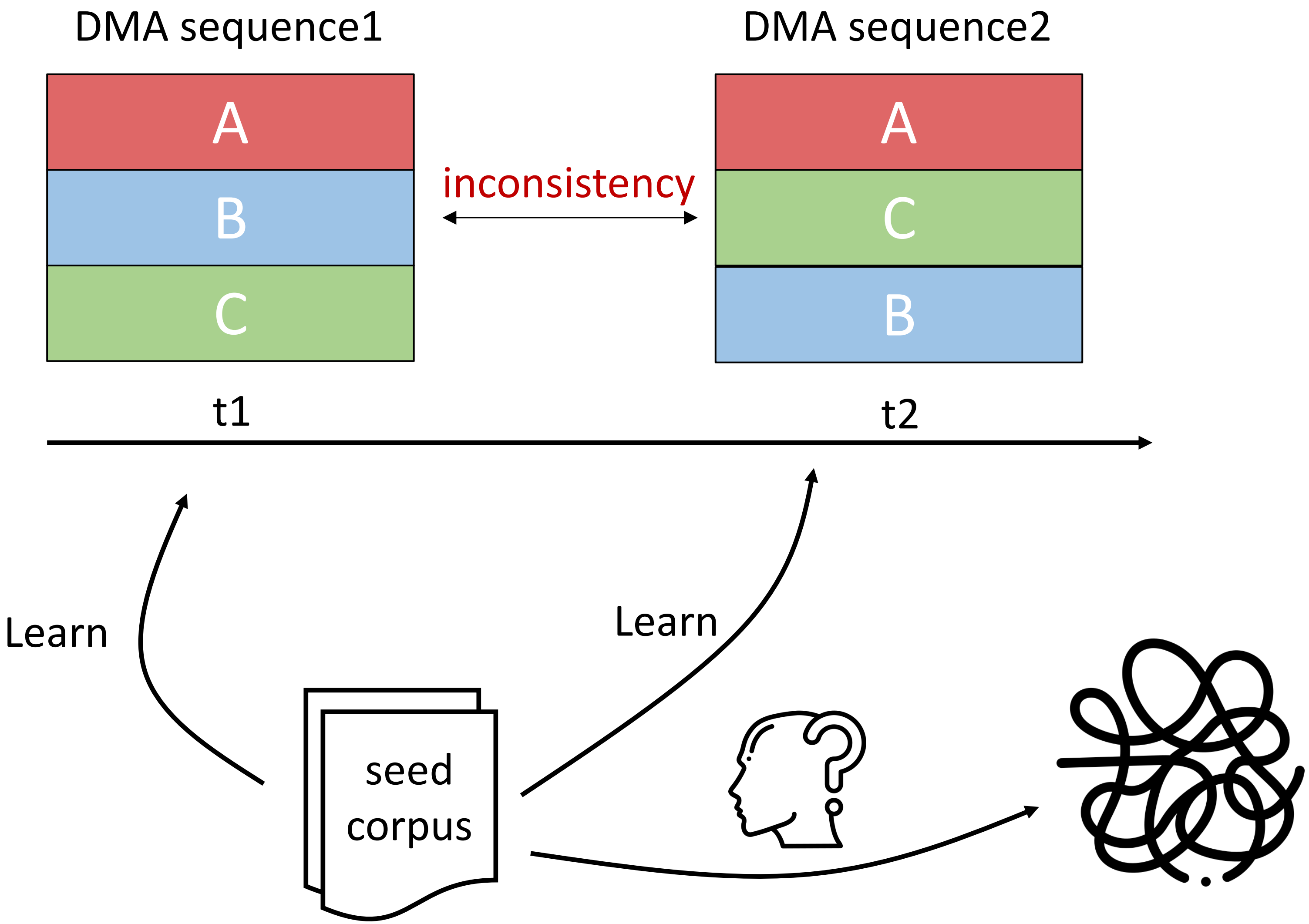
Eliminate the **pointer** in each node while leaving the structure's **semantics** intact



Nested DMA Structures

Flattened DMA Structures

Recall DMA Feature2: Dynamic Node Type



Fine-grained node-level **semantics** is required for **coverage-guided fuzzing**

2. Semantics-Aware Fuzzing via Seedpools

➢ Static Analysis to Label DMA Objects

DMA Object

```
void uhci_process_frame ( … ) {
    UHCI_QH qh;
    …
    if (is_qh) {
        pci_dma_read (&qh, sizeof(qh));
    }

    UHCI_TD td;
    …
    uhci_read_td (&td);

    uhci_handle_td (…);
    …
}
```

```
void uhci_read_td (UHCI_TD *td) {
    pci_dma_read (td, sizeof(*td));
    …
}
```

```
void uhci_handle_td ( … ) {
    UHCI_TD last_td;
    …
    uhci_read_td (&last_td);
    …
}
```

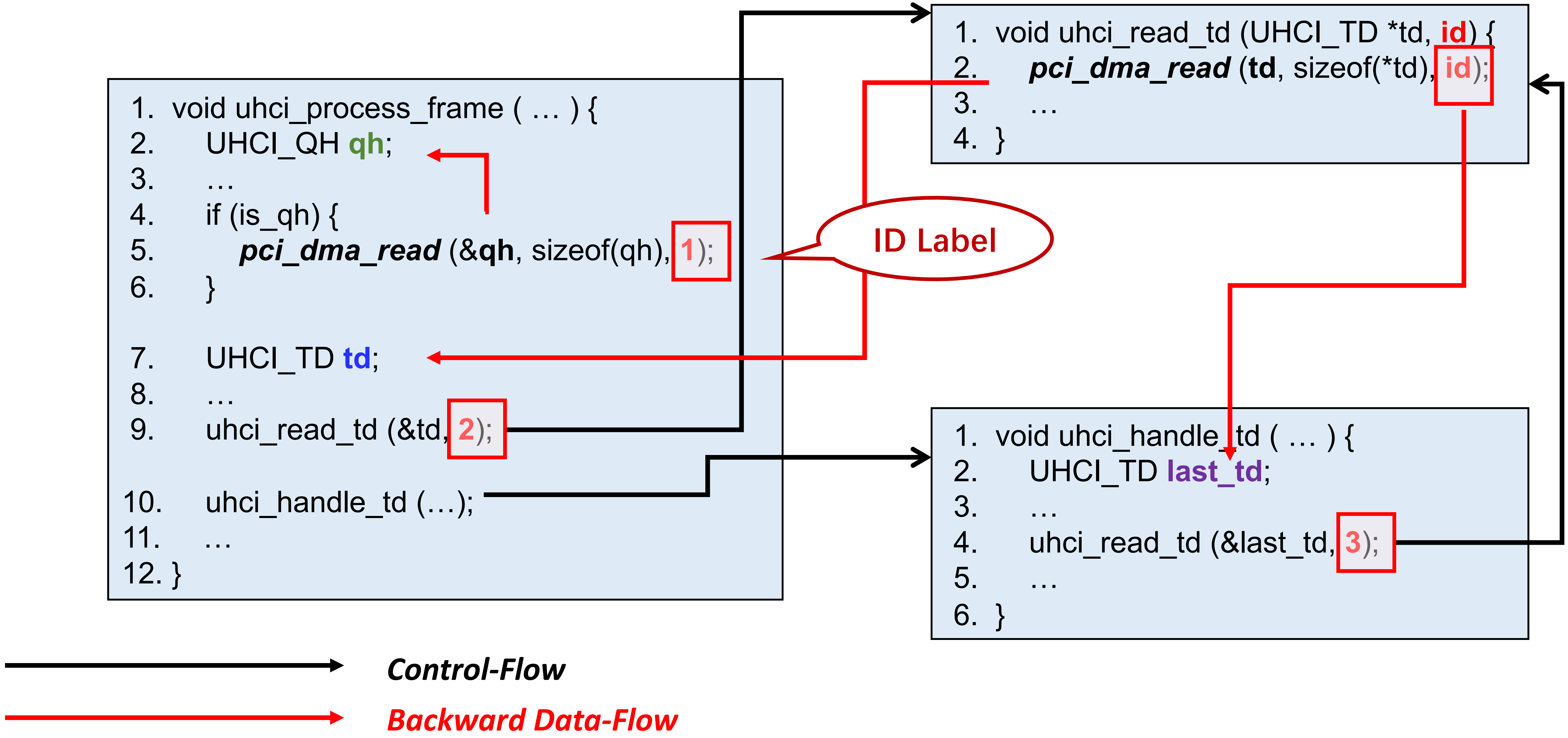
❶ ❷ ❸

Control-Flow

Backward Data-Flow

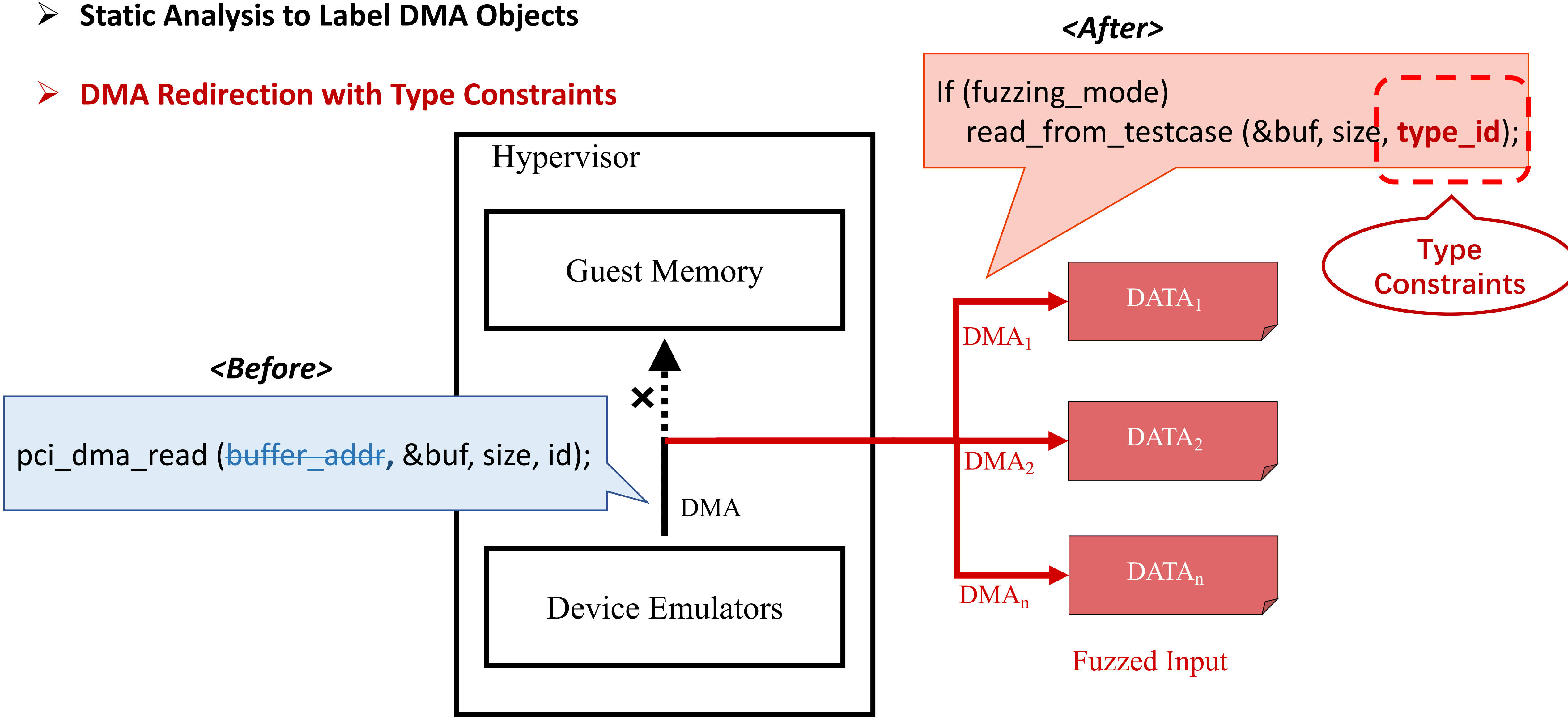
2. Semantics-Aware Fuzzing via Seedpools

- **Static Analysis to Label DMA Objects**



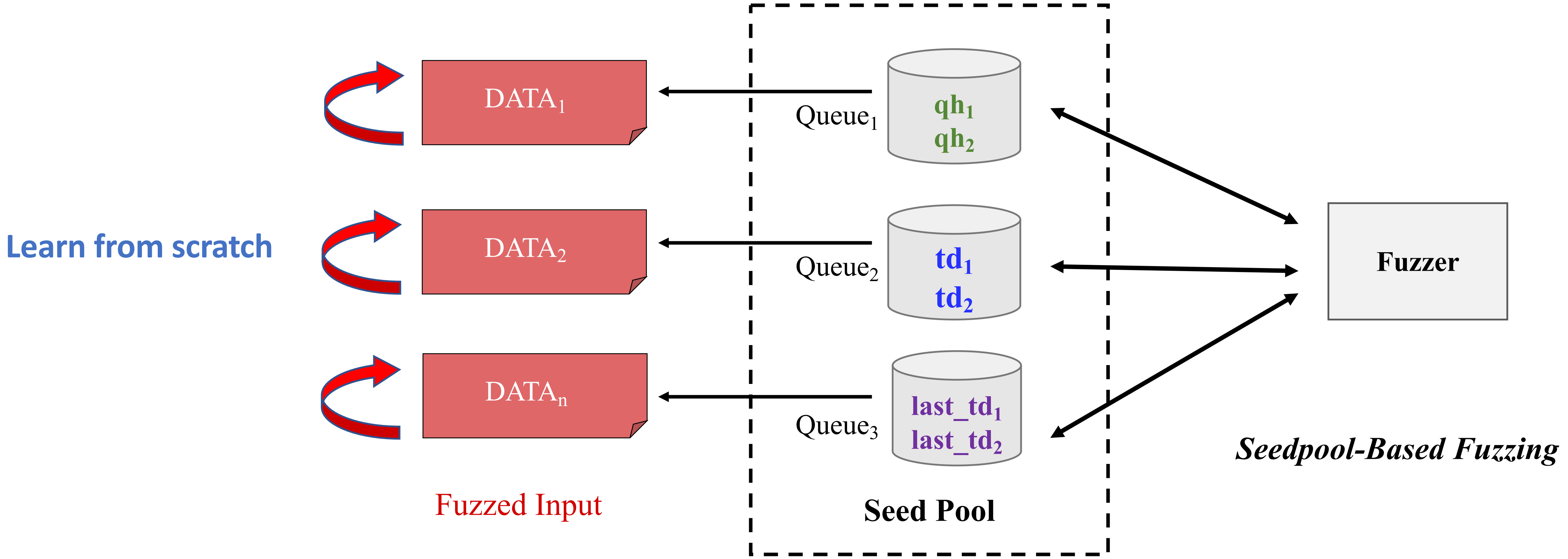
2. Semantics-Aware Fuzzing via Seedpools

- **Static Analysis to Label DMA Objects**
- **DMA Redirection with Type Constraints**



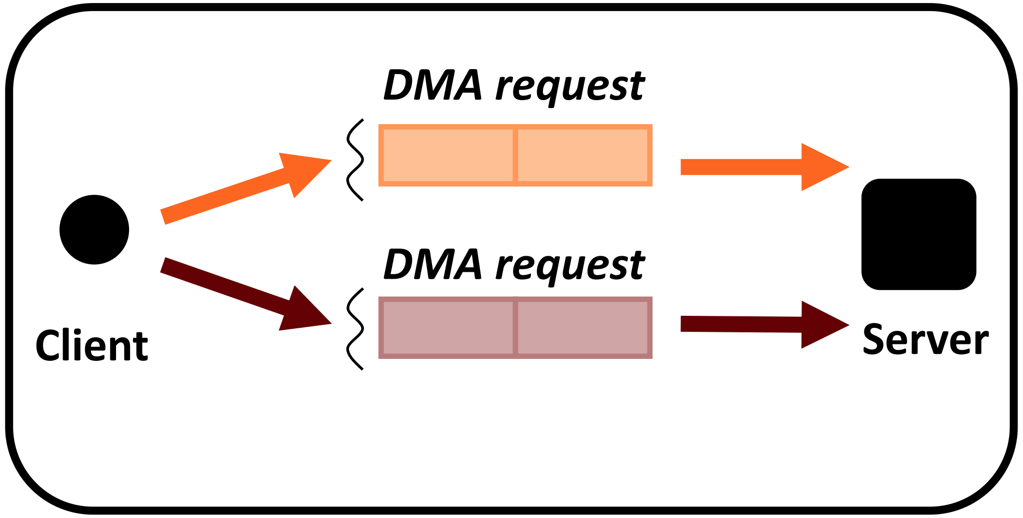
2. Semantics-Aware Fuzzing via Seedpools

- Static Analysis to Label DMA Objects
- DMA Redirection with Type Constraints
- Seedpool-Based Fuzzer Design



2. Semantics-Aware Fuzzing via Seedpools

- ➢ Static Analysis to Label DMA Objects
- ➢ DMA Redirection with Type Constraints
- ➢ Seedpool-Based Fuzzer Design
- ➢ **Semantics-aware Fuzzing Process**



Algorithm 1 Main semantics-aware fuzzing loop of V-Shuttle

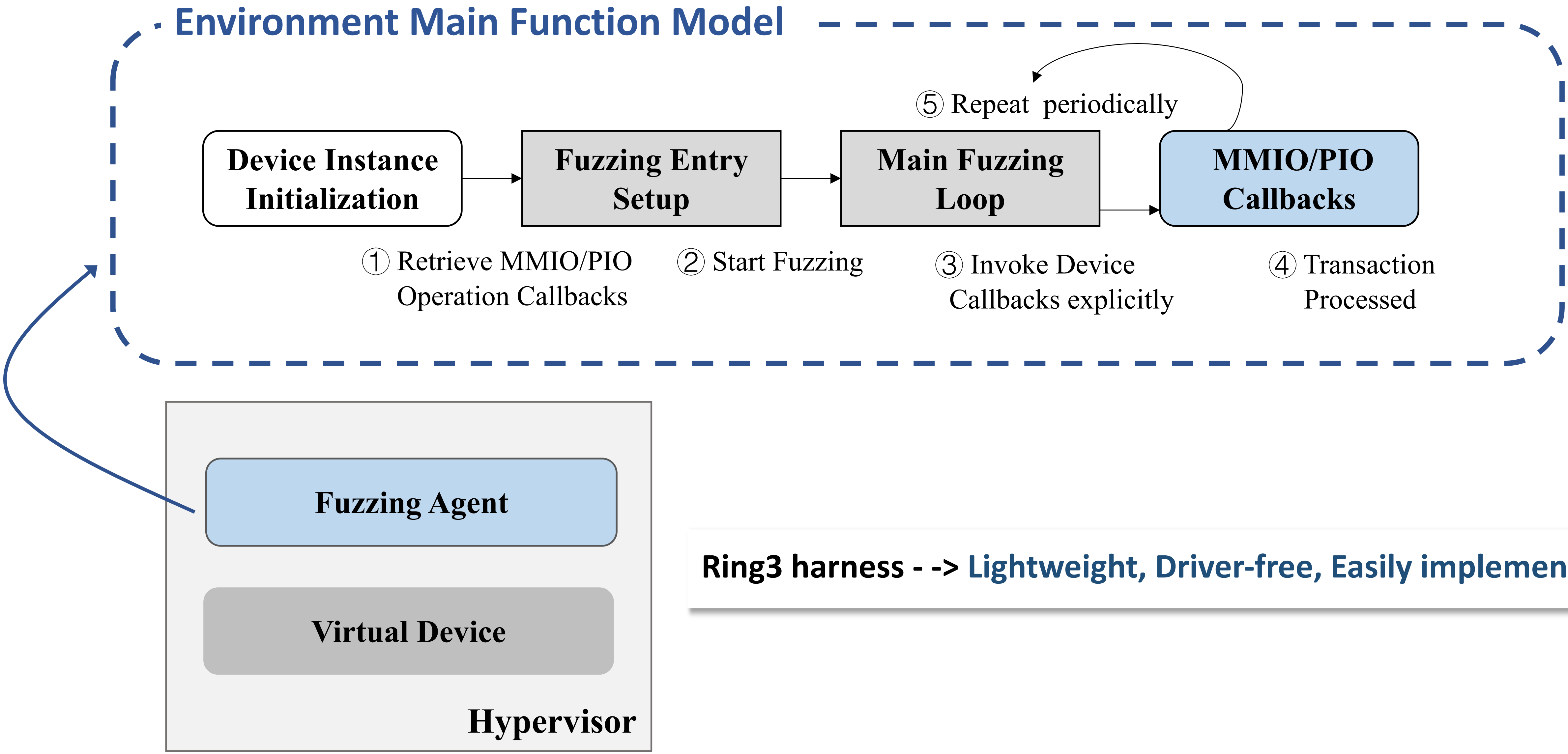
Input: Initial seeds queues *Seedpool*[], Target Hypervisor *H*

```
// setup each basic seed queues and global information ;
for all queue of the Seedpool[] do
    queue.setup();
end for
GlobalMap.init();
repeat
    id = H.request()
    seed = Mutate(Seedpool[id]);
    Cover = H.feed(seed);
    if Cover.haveNewCoverage() then
        Seedpool[id].push(seed)
    end if
until timeout or abort-signal;
```

Output: Crashing seeds *crashes*

1. Initialize
2. Wait for request
3. Mutate
4. Execute

3. Lightweight Fuzzing Loop



Ring3 harness - -> Lightweight, Driver-free, Easily implemented

Evaluations

Experiment Settings

➢ Experiment settings

- Two well-known hypervisors: QEMU 5.1.0, VirtualBox 6.1.14
- Build with ASAN to discover bugs
- Gcov-based coverage measurement
- Each hypervisor instance is tested for 24 hours

Scalability

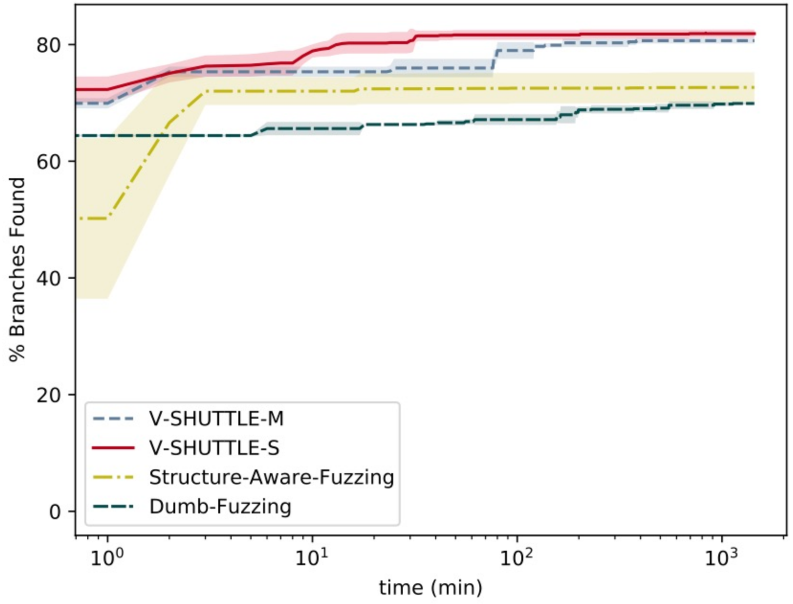
- Code coverage on ***16*** popular QEMU devices: Audio, Graphics, Network, USB, Storage
- Our solution has ***tolerable overhead*** as compared to the traditional dumb fuzzing

	Device	Line Coverage		Functions Coverage		Branches Coverage		Speed(exec/s)	
		Initial	Total	Initial	Total	Initial	Total	Dumb-Fuzzing	V-SHUTTLE
Audio	CS4231a	30.00%	**96.10%**	57.10%	**100.00%**	3.00%	**85.80%**	10918.21	**7632.70**
	Intel-HDA	68.30%	**95.00%**	78.60%	**95.20%**	42.10%	**78.30%**	9596.41	**8568.50**
	ES1370	54.20%	**99.62%**	73.70%	**100.00%**	33.80%	**91.91%**	8786.85	**6496.04**
	SoundBlaster	12.30%	**99.19%**	28.60%	**100.00%**	3.00%	**81.52%**	5123.76	**3242.22**
Graphics	ATI-VGA	27.40%	**86.00%**	66.70%	**80.00%**	15.30%	**79.40%**	10350.61	**10103.42**
Network	E1000	36.20%	**94.20%**	46.90%	**96.90%**	16.10%	**74.50%**	5532.90	**1186.92**
	NE2000	6.70%	**89.60%**	28.60%	**100.00%**	3.80%	**71.90%**	12213.31	**11392.45**
	PCNET	24.60%	**97.40%**	44.80%	**100.00%**	8.30%	**88.90%**	5880.21	**4833.35**
	RTL8139	28.10%	**97.60%**	59.10%	**97.70%**	12.30%	**88.40%**	6333.37	**5495.18**
USB	UHCI	81.30%	**89.10%**	86.10%	**88.90%**	68.90%	**82.30%**	10592.12	**9273.25**
	EHCI	40.70%	**82.70%**	53.40%	**89.00%**	32.70%	**71.90%**	3869.43	**2265.34**
	OHCI	46.90%	**83.70%**	65.10%	**86.00%**	33.30%	**79.20%**	7221.49	**5228.43**
Storage	NVME	38.60%	**72.40%**	47.30%	**76.40%**	22.80%	**65.10%**	10981.52	**7870.23**
	Lsi53c895a	26.90%	**79.00%**	46.70%	**71.10%**	9.30%	**75.70%**	6363.84	**4091.53**
	Megasas	58.10%	**63.80%**	68.30%	**70.00%**	43.90%	**58.50%**	5863.47	**4558.58**
	AHCI	75.30%	**81.80%**	78.60%	**82.10%**	51.90%	**61.60%**	5577.74	**5525.55**
Average		40.98%	**87.95%**	58.10%	**89.58%**	25.03%	**77.18%**	7844.64	**6110.23**

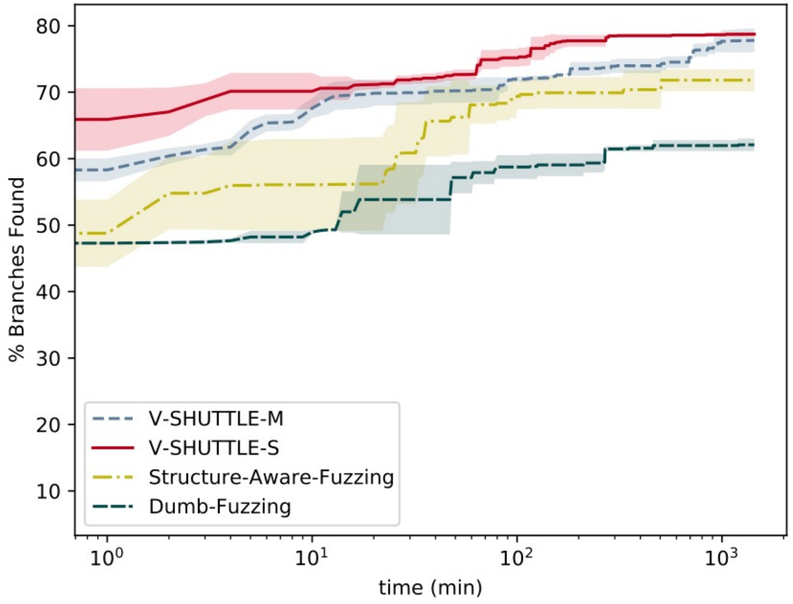
Code Coverage Enhancement

- Comparison of Dumb Fuzzing, Structure-Aware Fuzzing, V-SHUTTLE Main Framework, V-SHUTTLE with Semantics-Aware Fuzzing Mode

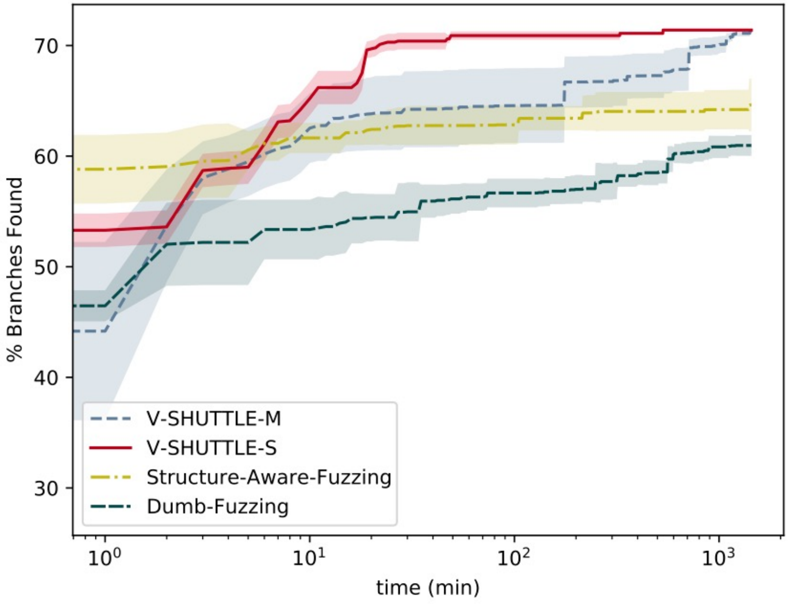
- V-SHUTTLE performs better with the semantics-aware fuzzing mode



(a) UHCI



(b) OHCI



(c) EHCI

Code Coverage Enhancement

- Compared with state-of-the-art hypervisor fuzzers
  - *VDF [RAID'17], Hyper-Cube [NDSS'20], Nyx [Sec'21]*
- V-SHUTTLE presents coverage improvement over the others

	VDF	Hyper-Cube	Nyx	V-Shuttle		
Device	Cov	Cov	Cov	Cov	Std	△
CS4231a	56.00%	74.76%	74.76%	**85.80%**	1.07	11.04%
Intel-HDA	58.60%	79.17%	78.33%	78.30%	0.55	-0.03%
ES1370	72.70%	91.38%	91.38%	91.91%	1.21	0.54%
SoundBlaster	81.00%	83.80%	81.34%	81.52%	0.42	0.18%
E1000	81.60%	66.08%	54.55%	**74.50%**	0.90	19.95%
NE2000	71.70%	71.89%	71.89%	71.90%	0.92	0.01%
PCNET	36.10%	78.71%	89.49%	88.90%	1.35	-0.59%
RTL8139	63.00%	74.68%	79.28%	**88.40%**	0.64	8.72%

Vulnerability Discovery

- Discovered new vulnerabilities
  - ***35 new vulnerabilities*** found in QEMU and VirtualBox with ***17 CVE assigned***
  - UAF, Integer overflow, OOB access, etc., including high-impact ***exploitable*** vulnerabilities
- Reasonable time to rediscover previously known vulnerabilities

Bug	Description	Exec	Time	Found
CVE-2020-25625	OHCI infinite loop	40.5M	2 hrs, 16 min, 50 sec	✓
CVE-2020-25085	SDHCI Heap buffer overflow	8.88M	26 min, 19 sec	✓
CVE-2021-20257	E1000 inifinite loop	235k	40 sec	✓
CVE-2020-25084	EHCI use-after-free	79.4M	4 hrs, 37 min, 22 sec	✓
CVE-2020-11869	ATI-VGA integer overflow	35.6M	2 hrs, 22 min, 40 sec	✓

Hypervisor	Description	Device Type	CVE/Issue-ID	CVSS Score	Impact
QEMU	Heap buffer overflow (write) in ohci_copy_iso_td	USB	**CVE-2020-25624**	5.0	DoS
	Stack buffer overflow (read) in ohci_service_iso_td	USB	confirmed	-	DoS
	Heap buffer overflow (read) in ohci_service_td	USB	confirmed	-	DoS
	Infinite loop in e1000e_write_packet_to_guest	Network	CVE-2020-25707	2.5	DoS
	OOB access in ati_2d_blt	Graphics	CVE-2020-27616	2.8	DoS
	Reachable assert failure via eth_get_gso_type	Network	CVE-2020-27617	3.8	DoS
	Divide by zero in dwc2_handle_packet	USB	CVE-2020-27661	3.8	DoS
	Integer Overflow in sm501_2d_operation	Graphics	requested	-	DoS
	Infinite loop in xhci_ring_chain_length	USB	CVE-2020-14394	3.2	DoS
	Heap-use-after-free in nic_reset	Network	requested	-	Exploitable
	Heap buffer overflow (write) in dp8393x_do_transmit_packets	Network	confirmed	-	DoS
	Failed malloc in omap_rfbi_transfer_start	Graphics	requested	-	DoS
	Infinite loop in allwinner_sun8i_emac_get_desc	Network	confirmed	-	DoS
	Divide by zero in exynos4210_ltick_cnt_get_cnto	Timer	confirmed	-	DoS
	Divide by zero in zynq_slcr_compute_pll	Misc	confirmed	-	DoS
	Failed malloc in vmxnet3_activate_device	Network	CVE-2021-20203	3.2	DoS
	NULL pointer derefence in fdctrl_read	Storage	CVE-2021-20196	3.2	DoS
	Heap-use-after-free in ehci_flush_qh	USB	requested	-	Exploitable
	NULL pointer derefence in lsi53c895a	Storage	requested	-	DoS
	NULL pointer derefence in vmport_ioport_read	Core	requested	-	DoS
	NULL pointer derefence in a9_gtimer_get_current_cpu	Timer	requested	-	DoS
	Assertion in usb_msd_send_status	USB	#1901981	-	DoS
	Assertion in usb_ep_get	USB	#1907042	-	DoS
	Assertion in ohci_frame_boundary	USB	#1917216	-	DoS
	Assertion in vmxnet3_io_bar1_write	Network	#1913923	-	DoS
	Assertion in lsi_do_dma	Storage	#1905521	-	DoS
VirtualBox	Heap buffer overflow (write) in xhciR3WriteEvent	USB	CVE-2020-2905	8.2	Exploitable
	Heap buffer overflow (write) in xhciR3WriteEvent	USB	CVE-2020-14872	8.2	Exploitable
	OOB Read in ehciR3ServiceQHD	USB	CVE-2020-14889	6.0	Info leak
	Divide by zero in e1kTxDLoadMore	Network	CVE-2020-14892	5.5	DoS
	Integer overflow in e1kGetTxLen	Network	CVE-2021-2073	4.4	DoS
	Heap buffer overflow (write) in buslogicRegisterWrite	Storage	**CVE-2021-2074**	8.2	Exploitable
	Divide by zero in ataR3SetSector	Storage	CVE-2021-2086	6.0	DoS
	NULL pointer derefence in blk_read	Storage	CVE-2021-2130	4.4	DoS
	Unintialized stack object in LsiLogicSCSI	Storage	CVE-2021-2123	3.2	Info leak

Case Study – CVE-2020-25624

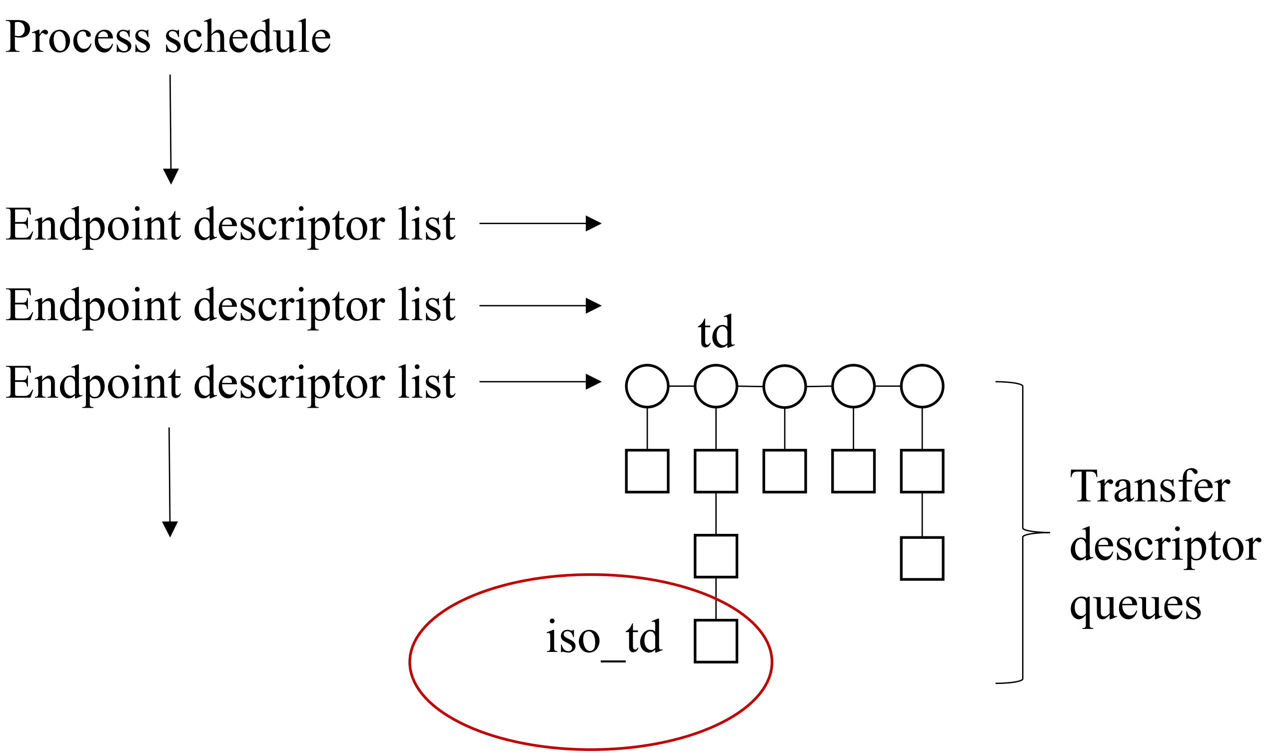
QEMU: *USB-OHCI* Out-of-Bounds Access

```
static int ohci_service_iso_td(OHCIState *ohci, struct ohci_ed *ed,
                               int completion)
{

    if (ohci_read_iso_td(ohci, addr, &iso_td)) {
        trace_usb_ohci_iso_td_read_failed(addr);
        ohci_die(ohci);
        return 1;
    }

    if ((start_addr & OHCI_PAGE_MASK) ≠ (end_addr & OHCI_PAGE_MASK)) {
        len = (end_addr & OHCI_OFFSET_MASK) + 0x1001
            - (start_addr & OHCI_OFFSET_MASK);
    } else {
        len = end_addr - start_addr + 1;
    }

    if (len && dir ≠ OHCI_TD_DIR_IN) {
        if (ohci_copy_iso_td(ohci, start_addr, end_addr, ohci→usb_buf,
len,                                 DMA_DIRECTION_TO_DEVICE)) {
            ohci_die(ohci);
            return 1;
        }
    }

}
```

reading iso_td

vulnerable point

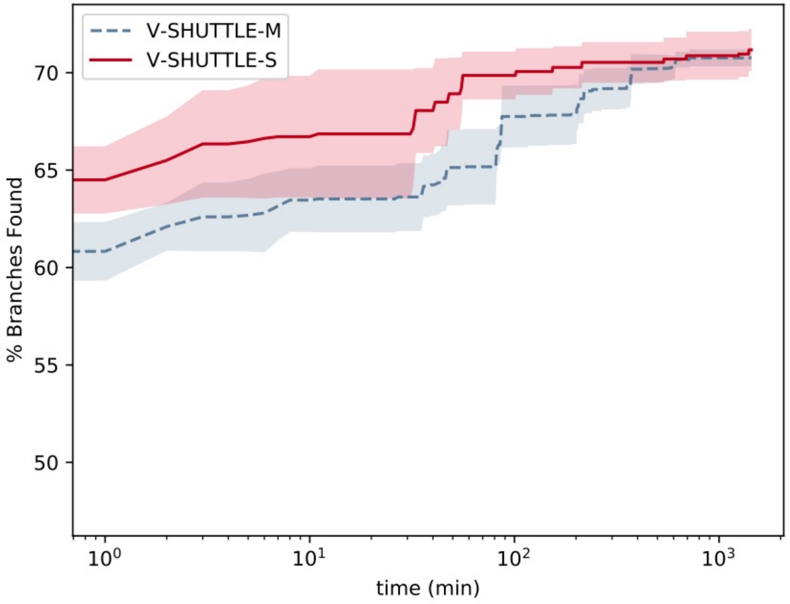
crash point



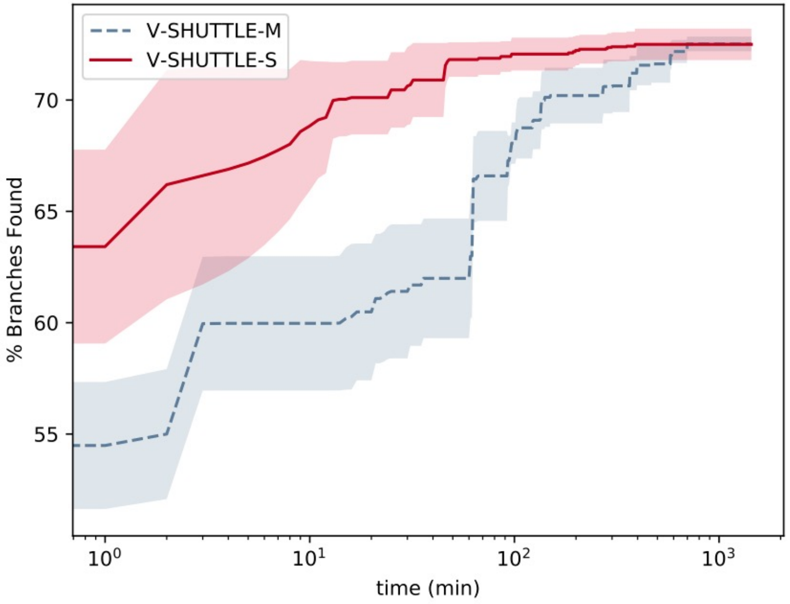
Deployment and Application

- V-SHUTTLE performs better with the semantics-aware fuzzing mode
- V-SHUTTLE's can be ported to Ant Group's commercial platform with little efforts
  - Lightweight: Takes about an hour to implement V-SHUTTLE into a new hypervisor via static analysis, some simple configurations and instrumentation

蚂蚁集团
ANT GROUP



(a) UHCI



(b) EHCI

Discussion

➢ Limitation and future work

- Automatic PoC reconstruction under persistent fuzzing.
- Supporting closed-source hypervisors by applying binary analysis technique.
- Fine-grained awareness of hypervisor internal states.

Conclusion

- We systematically study the driver-device interaction in virtual machine transaction and reveal that the data structures transferred via DMA have nested features.
- The first hypervisor fuzzer that automatically handles nested structures by semantics-aware DMA redirection
- Discovered 35 vulnerabilities with 17 CVEs assigned, and presented the better code coverage, compared to state-of-the-arts
- V-SHUTTLE: https://github.com/hustdebug/v-shuttle

Q&A

pgn@zju.edu.cn