面向Linux外设的虚拟化关键技术研究

博士论文答辩人：刘强

学院（系）：计算机科学与技术学院

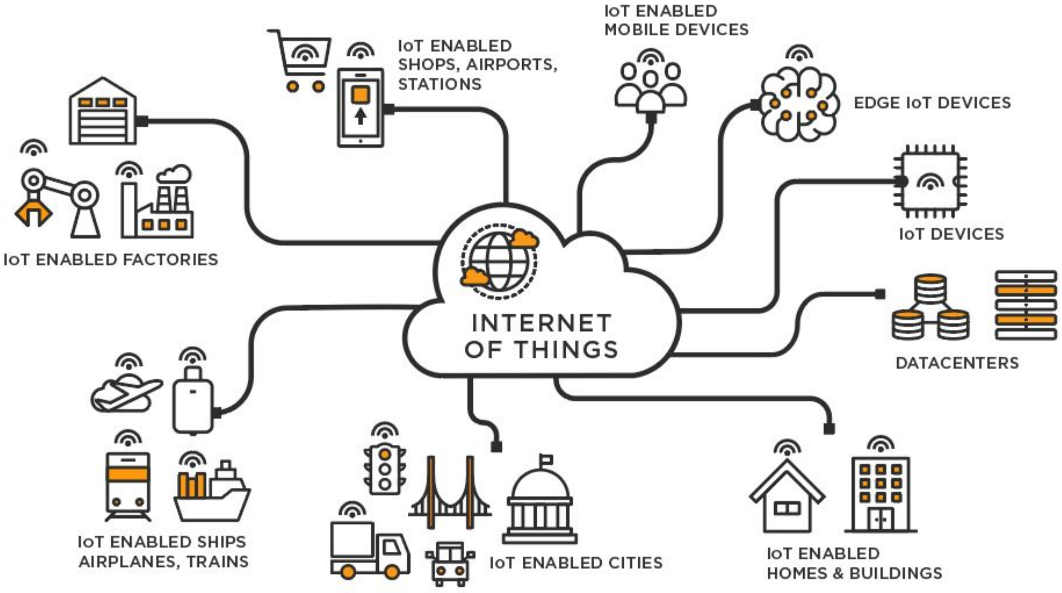
学科（专业）：网络空间安全

指导教师：周亚金

答辩时间：2023年9月1日

Linux物联网设备的安全性亟待分析和加强

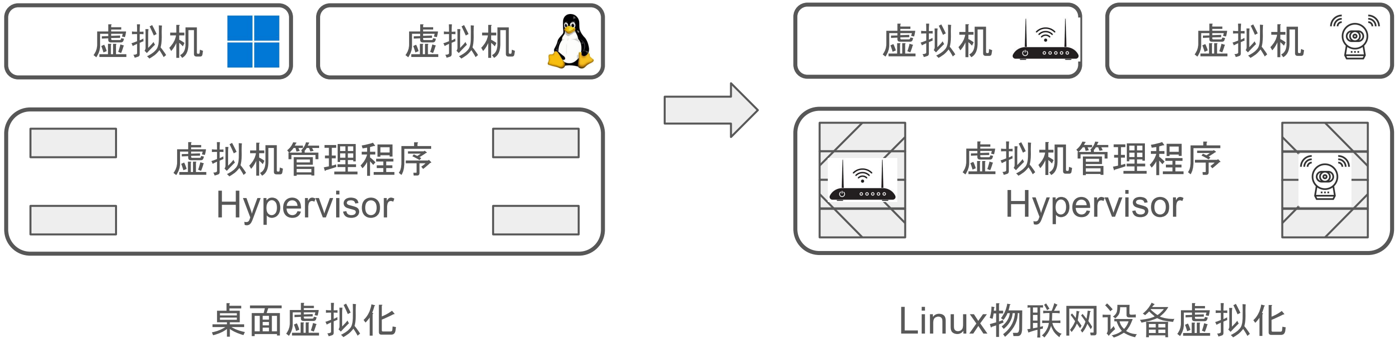
- 物联网(Internet of Things – IoT)应用广泛
- Linux物联网设备占比大，安全风险高(弱密码、未打补丁的软件)



1,200,000 Linux物联网设备
受Mirai Botnet影响

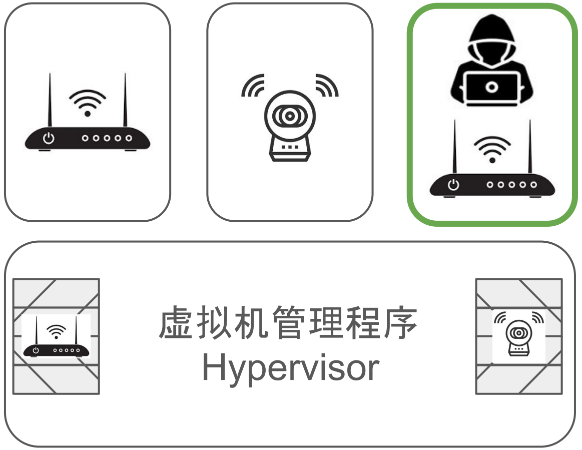
Linux物联网设备虚拟化的必要性

- 动态分析：如模糊测试、漏洞分析和利用开发、大规模蜜罐部署等
- 难以部署：因为真实设备难获取、扩展性差、可调式性差



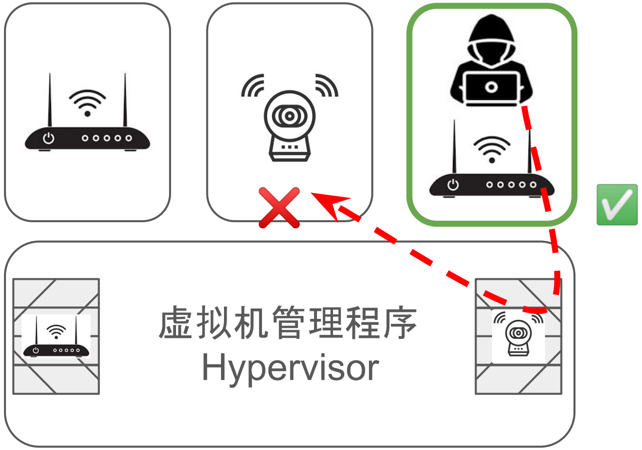
Linux物联网设备虚拟化的核心目标

- 保真性(fidelity):确保虚拟的物联网设备与物理的物联网设备一致
- 安全性(security):确保各个虚拟的物联网设备之间不互相影响



Linux物联网设备虚拟化的核心目标

- 保真性(fidelity):确保虚拟的物联网设备与物理的物联网设备一致
- 安全性(security):确保各个虚拟的物联网设备之间不互相影响



面向Linux物联网设备的虚拟化关键技术研究

基于模型引导内核执行的
虚拟执行环境构建研究

FirmGuide, ASE’21

基于依赖感知消息模型的
虚拟执行环境模糊测试研究

ViDeZZo, S&P’23

面向Linux物联网设备的虚拟化关键技术研究

基于模型引导内核执行的
虚拟执行环境构建研究

FirmGuide, ASE’21

基于依赖感知消息模型的
虚拟执行环境模糊测试研究

ViDeZZo, S&P’23

保真性

安全性

FirmGuide: Boosting the Capability of Rehosting Embedded Linux Kernels through Model-Guided Kernel Execution

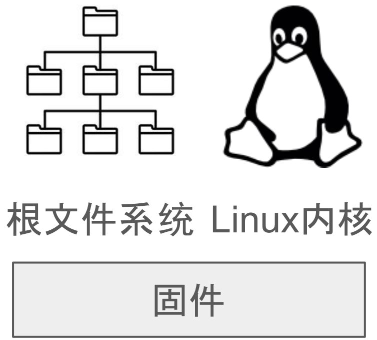
Qiang Liu[1*] Cen Zhang[2*] Lin Ma[1] Muhui Jiang[1,3] Yajin Zhou[1] Lei Wu[1] Wenbo Shen[1] Xiapu Luo[3] Yang Liu[2] Kui Ren[1]

[1]Zhejiang University [2]Nanyang Technological University [3]The Hong Kong Polytechnic University

[*]The first two authors contributed equally to this work.

如何重新托管Linux物联网设备的内核？

- Linux物联网设备固件由根文件系统和Linux内核组成
- 已有的虚拟执行环境构建工具(重新托管)只能处理根文件系统
- 不处理Linux内核，保真度低、也无法分析该Linux内核里的安全问题



观察1：
高保真I-型、低保真II-型外围设备即可启动Linux内核

- I-型外围设备：中断控制器，定时器，串口
- II-型外围设备：PCIE，存储设备，网络设备，音频设备，USB

```
/dts-v1/;
/ {
    compatible = "plxtech,nas782x";
    cpus { }; # processor
    memory { }; # memory
    gic@47001000 { }; # interrupt controler
    timer@44400200 { }; # timer
    uart@44200000 { }; # uart
    reset-controller@44E00034 { };
    rps@44400000 { };
    oscillator { };
    sysclk { };
    plla@44e001f0 { };
    pllb@44f001f0 { };
    stdclk { };
    twdclk { };
    gmacclk { };
    pcie-controller@47C00000 { };
    pcie-controller@47E00000 { };
    local-timer@47000600 { };
    watchdog@47000620 { };
    sata@45900000 { };
    nand@41000000 { };
    ethernet@40400000 { };
    ehci@40200100 { };
    leds { };
    };
};
```

```
void start_kernel(void)
{
    ...
    setup_arch(&command_line);
    vfs_caches_init_early();
    trap_init();
    mm_init();
    sched_init();
    early_irq_init();
    init_IRQ();
    tick_init();
    init_timers();
    hrtimers_init();
    softirq_init();
    timekeeping_init();
    time_init();
    sched_clock_postinit();
    local_irq_enable();
    console_init();
    sched_clock_init();
    calibrate_delay();
    vfs_caches_init(totalram_pages);
    proc_root_init();
    rest_init();
}
```

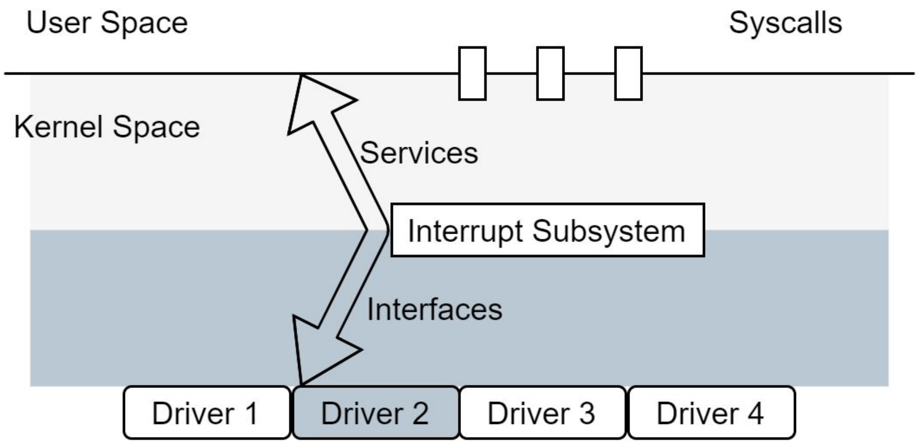
I-型	II-型


```
/dts-v1/;
/ {
    compatible = "plxtech,nas782x";
    cpus { }; # processor
    memory { }; # memory
    gic@47001000 { }; # interrupt controler
    timer@44400200 { }; # timer
    uart@44200000 { }; # uart
    reset-controller@44E00034 { };
    rps@44400000 { };
    oscillator { };
    sysclk { };
    plla@44e001f0 { };
    pllb@44f001f0 { };
    stdclk { };
    twdclk { };
    gmacclk { };
    pcie-controller@47C00000 { };
    pcie-controller@47E00000 { };
    local-timer@47000600 { };
    watchdog@47000620 { };
    sata@45900000 { };
    nand@41000000 { };
    ethernet@40400000 { };
    ehci@40200100 { };
    leds { };
    };
};
```

观察2:
Linux内核子系统定义状态机抽象外围设备行为

Linux内核中断子系统支持多中断控制器

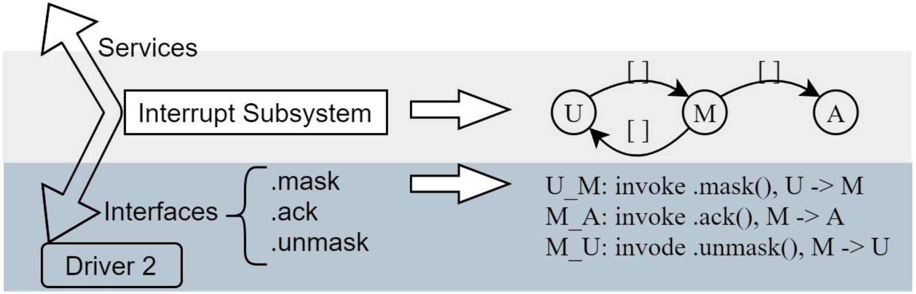
- ralink-rt2880-intc
- qca,ar7240-intc
- marvell,orion-intc
- marvell,orion-bridge-intc
- arm,cortex-a9-gic
- …



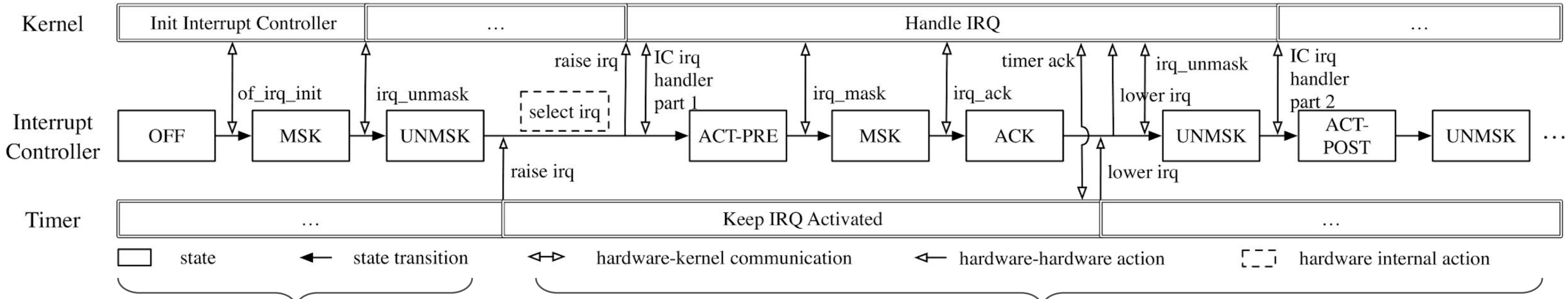
观察3：
状态机依赖 底层驱动的I/O读写序列 转移状态

.mask()
- MMIO Read A -> a (b’1000)
- a |= b’0001
- MMIO Write a (b’1001) -> A

.ack()
- MMIO Read B -> b (b’0001)
- switch(f(b)) …



状态机配合状态转移条件实现高保真I-型外围设备建模

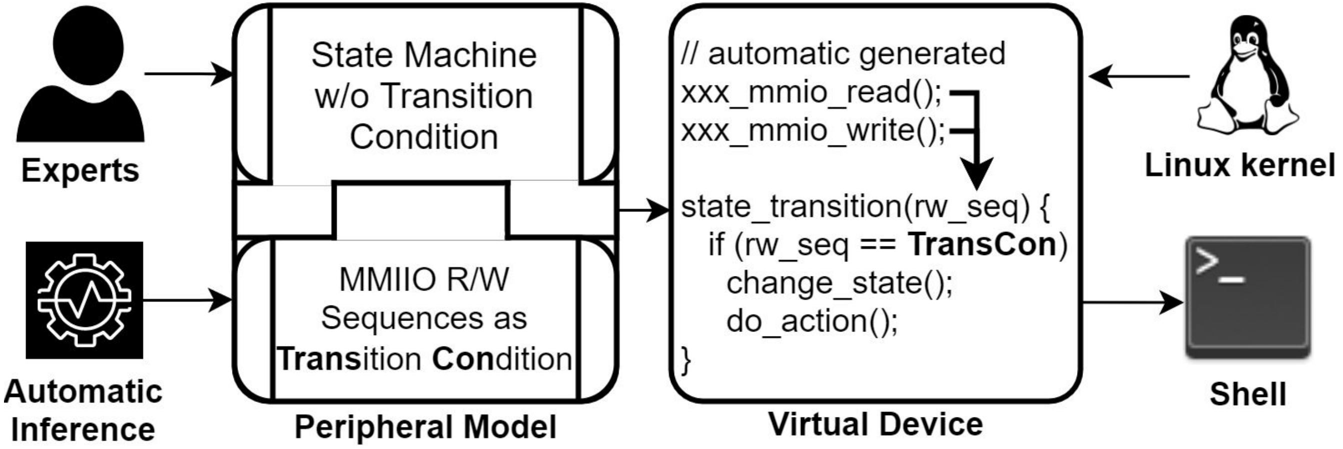


中断子系统 - 构建状态机

底层驱动 -> 推断自动状态转移条件

模型引导内核执行

- 虚拟外围设备=状态机（Linux内核子系统）+状态转移条件（I/O读写序列）



外围设备模型生成统计

- 9个I-型外围设备

Subtarget	Interrupt Controller	Timer
ramips/rt305x	ralink-rt2880-intc	`not necessary`
ath79/generic	qca,ar7240-intc	`not necessary`
kirkwood/generic	marvell,orion-intc marvell,orion-bridge-intc	marvell,orion-timer
bcm53xx/generic	arm,cortex-a9-gic	arm,cortex-a9-global-timer arm,cortex-a9-twd-timer
oxnas/generic	arm,arm11mp-gic	arm,arm11mp-twd-timer plxtech,nas782x-rps-timer

- 10/64个II-型外围设备（需要处理初始值的外围设备/总数）

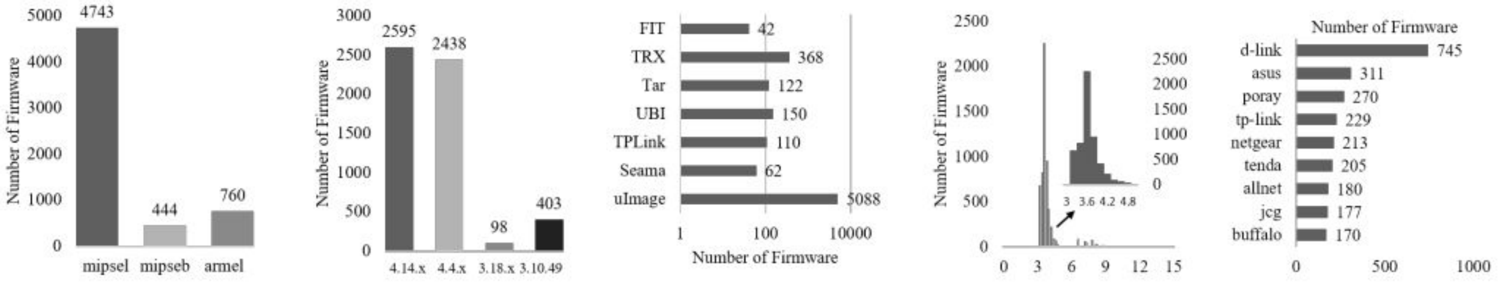
Subtarget	ramips/ rt305x	ath79/ generic	kirkwood/ generic	bcm53xx/ generic	oxnas/ generic
count	1/10	2/15	3/26	2/4	2/9

Linux物联网设备重新托管统计

- 重新托管了超过96%的Linux物联网设备的内核

SoC	Unpack	Kernel	Booting Validation	
			User Space	Shell
Overall	6,192	6,188	5947 (96.11%)	5469 (88.38%)

- 支持了2架构、22内核版本、7+固件格式、10+厂商、26片上系统等



(a) Architecture (b) Kernel Version (c) Firmware Format (d) Firmware Size (e) Top-10 Vendors

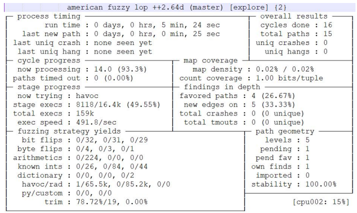
保真度测试及应用

- 自动化生成的模型 v.s. 专家编写的模型(Ground Truth)

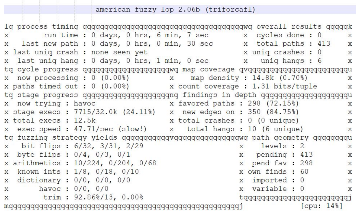
Models	Pass	Skipped	Failed	Total
Generated	1049	164	46	1259
Ground Truth	1049	164	46	1259

- 应用1:内核漏洞复现和漏洞利用开发
- 应用2:模糊测试

CVE ID	CVE Type	Status	Version
CVE-2016-5195	Race Condition	✗	N/A
CVE-2016-8655	Race Condition	✓ †	□
CVE-2016-9793	Integer Overflow	✓	□ ◊
CVE-2017-7038	Integer Overflow	✓ †	◊
CVE-2017-1000112	Buffer Overflow	✓ †	△
CVE-2018-5333	NULL Pointer Dereference	✓ †	□ ◊



UnicoreFuzz



TriforceAFL

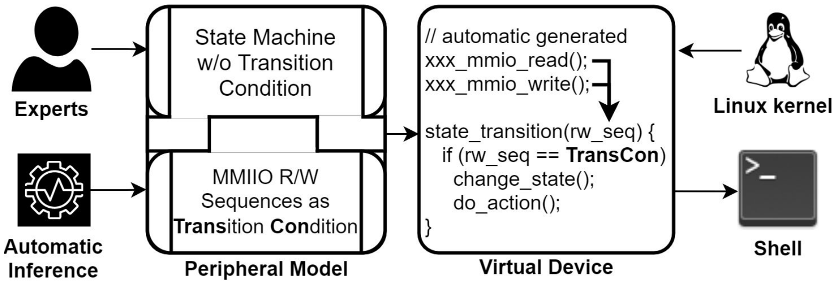
FirmGuide: Boosting the Capability of Rehosting Embedded Linux Kernels through Model-Guided Kernel Execution

如何重新托管Linux物联网设备的内核？

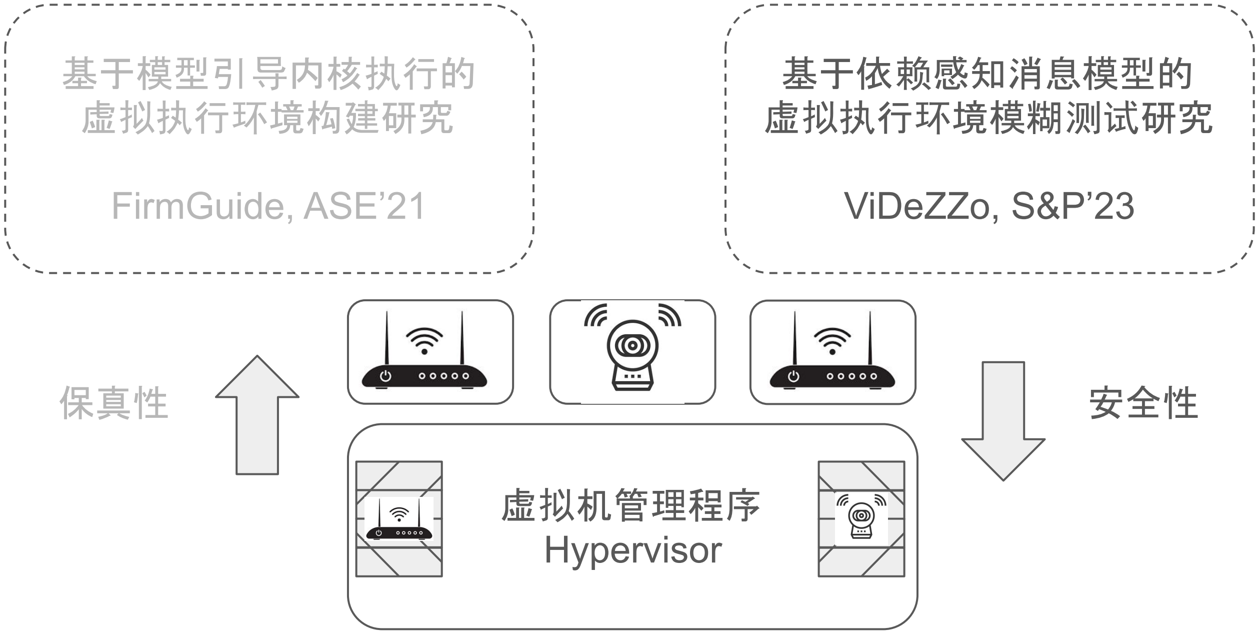
状态机配合状态转移条件实现高保真I-型外围设备建模

- 虚拟外围设备=状态机（Linux内核子系统）+状态转移条件（I/O读写序列）

II-型外围设备的建模技术仍需加强，进一步使能更多的应用



面向Linux物联网设备的虚拟化关键技术研究



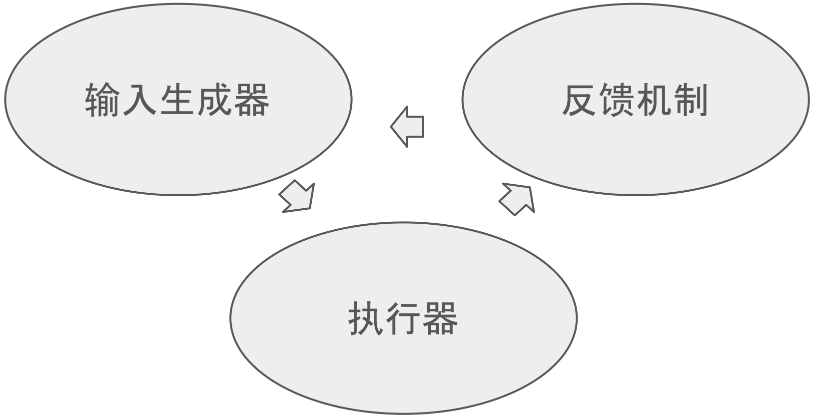
ViDeZZo: Dependency-aware Virtual Device Fuzzing

Qiang Liu (Zhejiang University; EPFL) Flavio Toffalini (EPFL)
Yajin Zhou (Zhejiang University) Mathias Payer (EPFL)

S&P 2023

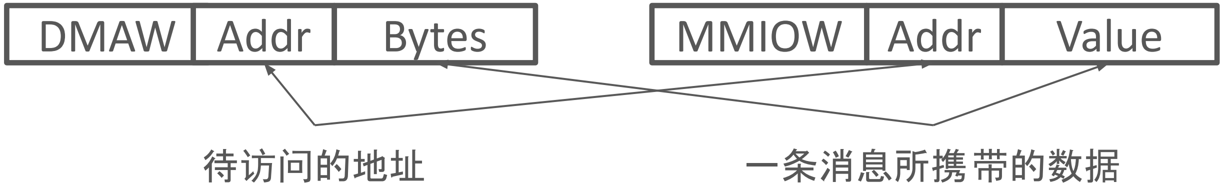
如何提高虚拟机管理程序模糊测试的效率？

- 模糊测试由输入生产器、执行器和反馈机制组成
- 已有的虚拟机管理程序模糊测试在执行器、反馈机制取得了进展
- 但受限于复杂的输入依赖，测试效率仍然较低



结构化的和按特定顺序的虚拟设备消息

- 输入/输出（I/O）
    - 内存映射I/O（Memory-Mapped I/O，MMIO）
    - 端口I/O（Port I/O，PIO）
    - 直接内存访问（Direct Memory Access，DMA）
- 时间调整

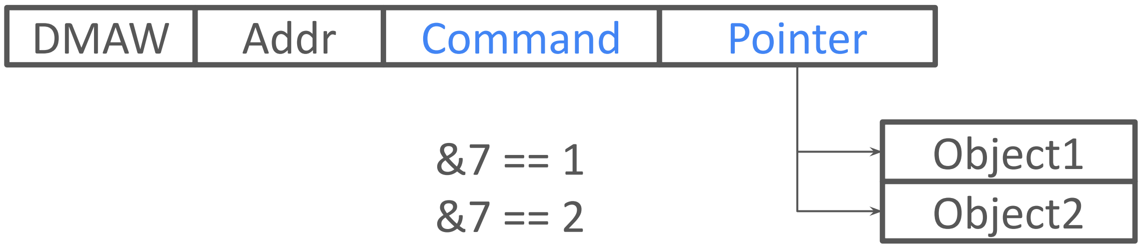


挑战1: 消息内依赖

- 虚拟设备消息中的一个字段可能依赖另一个字段

示例 1

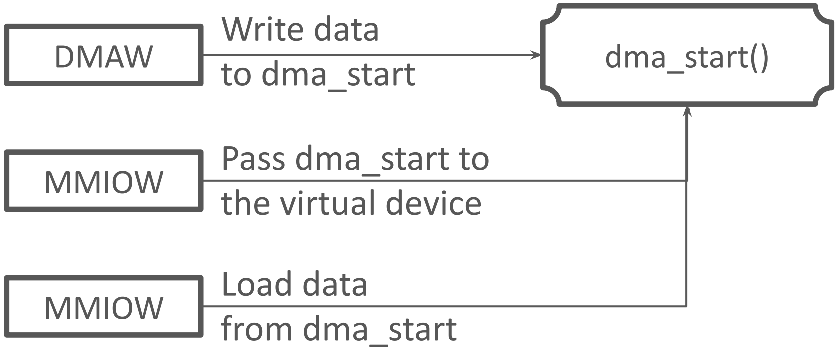
- Command的值不同, Pointer指向的对象不同



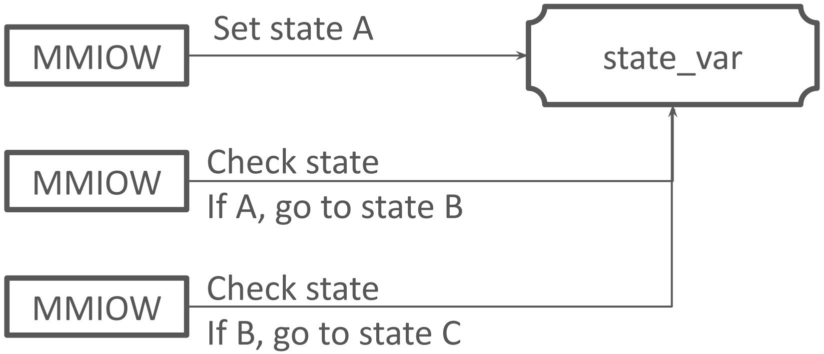
挑战2: 消息间依赖

- 一个虚拟设备消息可能依赖前一个消息

示例 2



示例 3



核心算法1:半自动消息内依赖注释提取

```
vd0=Model(‘tx’, 0)
vd0.add_struct(’tx_t’, {
 ‘command#0x4’: ’FLAG’,
 ‘pointer#0x4’: ’POINTER’})

vd0.add_point_to(‘tx_t.pointer’, [None,
’macaddr’, ’config’, None, None, None, None,
None], condition=[’tx_t.command.0’])
```

消息内依赖注释

⇦ 虚拟外围设备源代码

⇨ 支持消息内依赖的虚拟设备消息

核心算法2:多级消息突变器自动学习消息间依赖

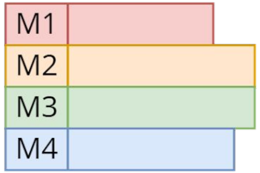
消息级	ChangeAddr, etc.
序列级	ShuffleMessages, etc.
组级	GroupMessage

多级消息突变器

原虚拟设备消息

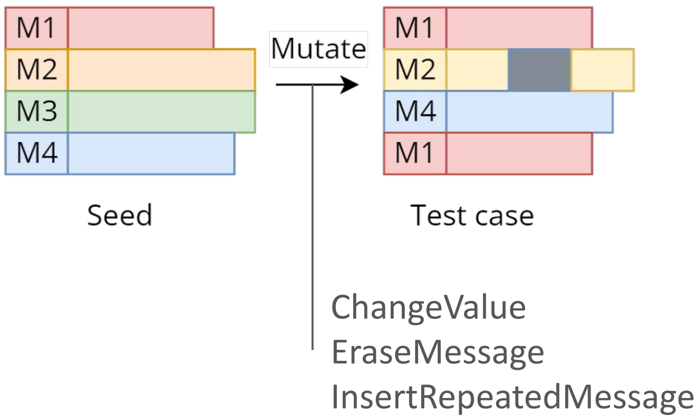
突变的虚拟设备消息(如提高了覆盖率,则保存在语料库中)

模糊测试工作流

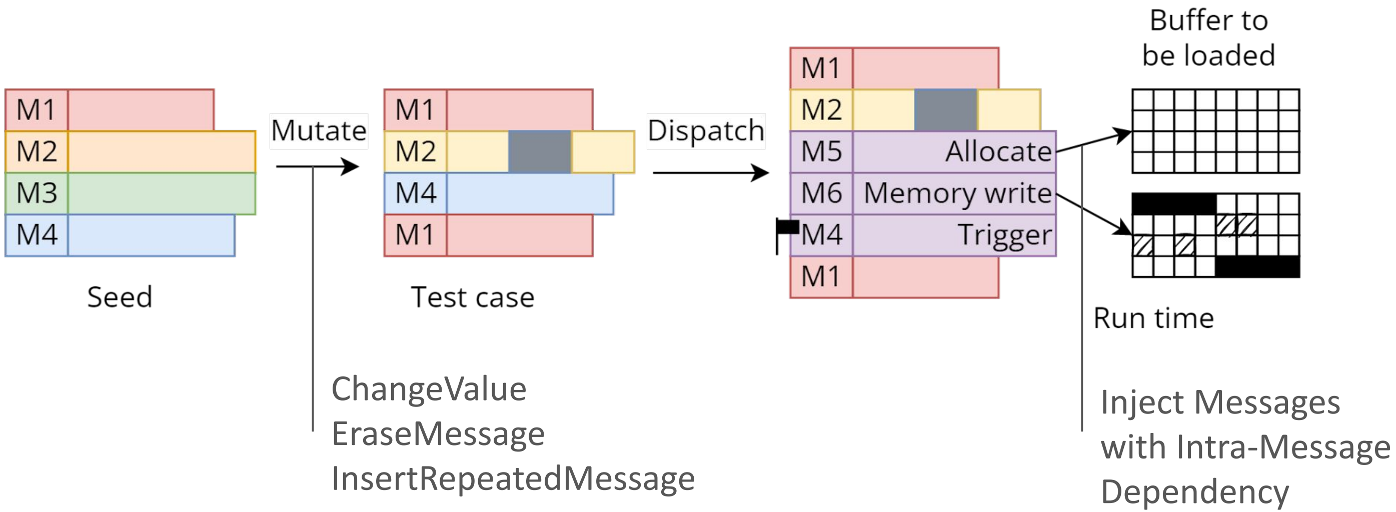


Seed

模糊测试工作流

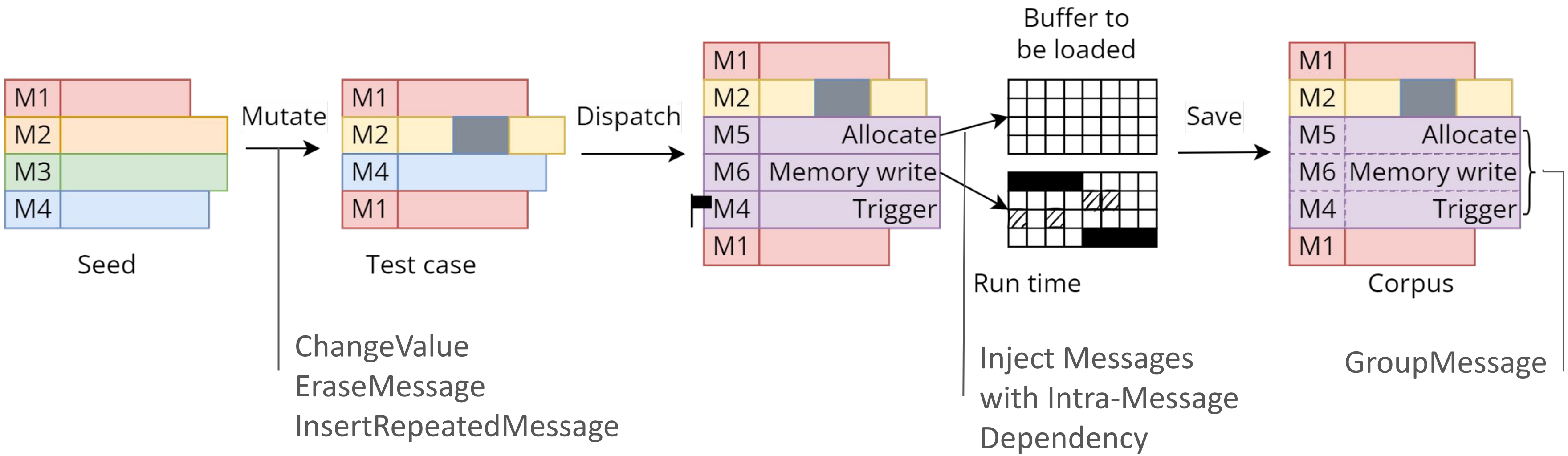

M1
M2
M3
M4
Seed
Mutate
M1
M2
M4
M1
Test case
ChangeValue
EraseMessage
InsertRepeatedMessage

模糊测试工作流



M1
M2
M3
M4
Seed
Mutate
M1
M2
M4
M1
Test case
ChangeValue
EraseMessage
InsertRepeatedMessage
Dispatch
M1
M2
M5 Allocate
M6 Memory write
M4 Trigger
M1
Buffer to be loaded
Run time
Inject Messages with Intra-Message Dependency

模糊测试工作流


M1
M2
M3
M4
Seed
Mutate
M1
M2
M4
M1
Test case
Dispatch
M1
M2
M5 Allocate
M6 Memory write
M4 Trigger
M1
Buffer to
be loaded
Run time
Save
M1
M2
M5 Allocate
M6 Memory write
M4 Trigger
M1
Corpus
ChangeValue
EraseMessage
InsertRepeatedMessage
Inject Messages
with Intra-Message
Dependency
GroupMessage

拓展性和效率

扩展性

- 支持28个虚拟设备
- 涵盖5种设备类型、4种架构
- 支持QEMU和VirtualBox

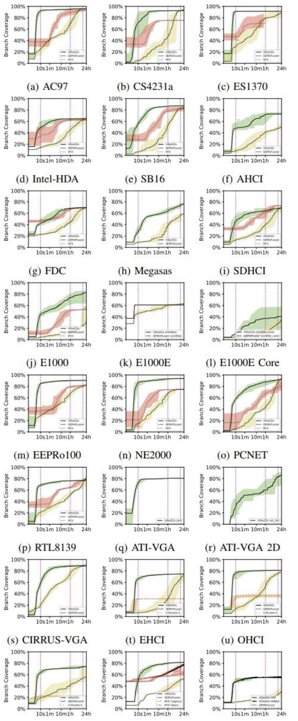
效率

- 更快地达到可比较的覆盖率
- 复现了24个漏洞、发现了28个新的漏洞

我们提交了多个补丁，有7个合并到QEMU中

Device	VDF	HYPERCUBE	Nyx-Legacy	V-SHUTTLE	QEMUFuzzer	VIDEZZ
QEMU-x86 Audio						
AC97	53.0%	100%	94.04%	—	**95.93%**	95.90%
CS4231a	56.0%	74.76%	75.36%	85.80%	**94.06%**	92.61%
ES1370	72.7%	91.38%	89.69%	91.91%	88.40%	**91.36%**
Intel-HDA	58.6%	79.17%	62.61%	78.30%	**65.87%**	64.78%
SB16	81.0%	83.80%	83.12%	81.52%	84.15%	**87.54%**
QEMU-x86 Storage						
AHCI	—	—	—	61.60%	49.89%	**62.06%**
FDC	70.5%	84.51%	**70.06%**	—	69.23%	69.72%
Megasas	—	—	—	58.50%	58.67%	**76.74%**
SDHCI	90.5%	81.15%	**73.58%**	—	71.34%	68.52%
VirtIO-BLK	—	—	—	—	30.55%	**55.39%**
QEMU-x86 Network						
E1000	81.6%	66.08%	53.36%	74.50%	35.32%	**82.27%**
E1000E (1/2)[1]	—	—	—	—	**63.12%**	60.94%
E1000E (2/2)[1]	—	—	—	—	35.48%	**40.84%**
EEPro100	75.4%	83.32%	82.12%	—	82.13%	**90.46%**
NE2000	71.7%	71.89%	74.35%	71.90%	75.09%	**94.00%**
PCNET	36.1%	78.81%	78.87%	88.90%	**93.27%**	92.10%
RTL8139	63.0%	74.68%	**83.33%**	80.82%	83.06%	77.46%
QEMU-x86 Display						
ATI-VGA (1/2)[2]	—	—	—	79.40%	—	80.69%
ATI-VGA (2/2)[2]	—	—	—	—	—	85.67%
CIRRUS-VGA	—	—	—	—	88.65%	**89.68%**
QEMU-x86 USB						
EHCI	—	—	—	31.19%	71.84%	**71.96%**
OHCI	—	—	—	36.62%	77.33%	**83.99%**
UHCI	—	—	—	22.27%	55.90%	**72.00%**
XHCI	—	64.40%	63.24%	—	52.92%	**81.63%**
			Nyx-Spec			
XHCI			77.12%			
QEMU-x86_64						
VirtIO-BLK	—	—	—	—	—	55.39%
QEMU-AArch32						
PL041 (Audio)	—	—	—	—	—	83.91%
SMC91C111 (Net)	—	—	—	—	92.14%	92.98%
TC6393XB (Display)	—	—	—	—	—	76.38%
QEMU-AArch64						
XLNX-ZYNQMP-CAN	—	—	—	—	—	70.42%
XLNX-DP (Display)	—	—	—	—	—	90.42%
VirtualBox x86_64						
SB16	—	—	—	—	—	61.33%
FDC	—	—	—	—	—	39.32%
PCNET	—	—	—	—	—	48.35%
OHCI	—	—	—	—	—	36.13%

[1] We collected the coverage in `e1000e.c` and `e1000e_core.c`, respectively.
[2] We collected the coverage in `ati.c` and `ati_2d.c`, respectively. V-SHUTTLE authors confirmed that they did n consider `ati_2d.c`. Therefore, it shows — in the table.



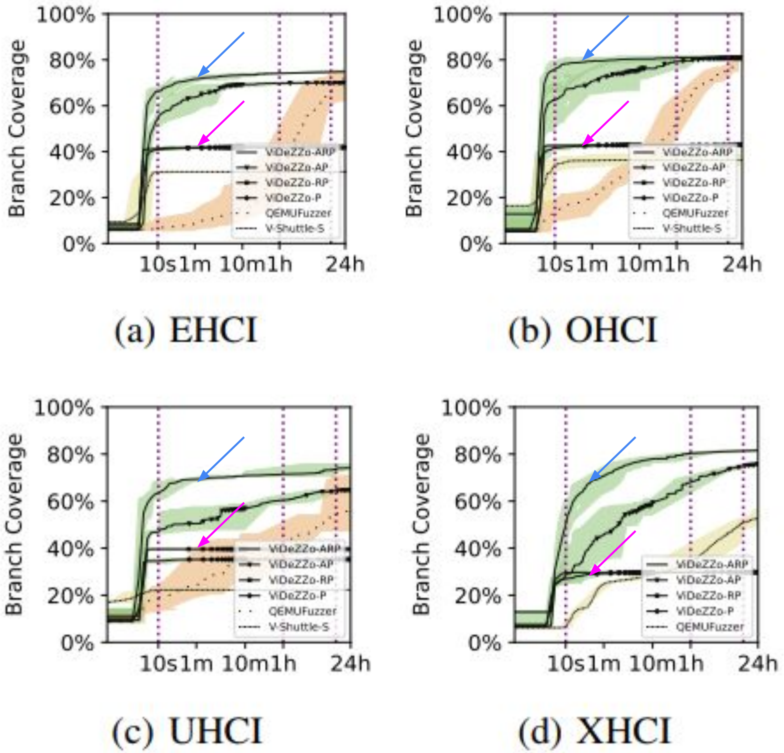
(a) AC97 (b) CS4231a (c) ES1370
(d) Intel-HDA (e) SB16 (f) AHCI
(g) FDC (h) Megasas (i) SDHCI
(j) E1000 (k) E1000E (l) E1000E Core
(m) EEPRo100 (n) NE2000 (o) PCNET
(p) RTL8139 (q) ATI-VGA (r) ATI-VGA 2D
(s) CIRRUS-VGA (t) EHCI (u) OHCI
(v) UHCI (w) XHCI (x) VirtIO-Blk

							1	An	Fixed
							2	An	Fixed
27	sb16	audio	qemu	6.1.50	i386	Assertion failure in audio_calloc() caused by sb16	N/A	An	Fixed
28	sb16	audio	qemu	6.1.50	i386	Abort in audio_calloc()	4	Vi	*Fixed(us)*
29	sdhci	storage	qemu	7.1.50	i386	Heap buffer overflow in sdhci_read_dataport()	9	QF	Fixed
30	smc91c111	net	qemu	7.1.93	aarch32	Out of bounds read/write in smc91c111	5	Vi	Open
31	tc6393xb	display	qemu	7.2.50	aarch32	negative-size-param in nand_blk_load_512()	23	Vi	Open
32	tc6393xb	display	qemu	7.2.50	aarch32	Heap buffer overflow in nand_blk_write_512()	7	Vi	Open
33	virtio-blk	storage	qemu	7.0.94	i386	Assertion failure in address_space_stw_le_cached()	5	An	Fixed
34	virtio-blk	storage	qemu	7.0.94	i386	Infinite loop in virtio_blk_handle_vq()	16	An	Fixed
35	vmxnet3	net	qemu	6.1.50	i386	Code should not be reached vmxnet3_io_bar1_write()	N/A	VS, Vi	Fixed
36	vmxnet3	net	qemu	6.1.50	i386	Three hw_error() in vmxnet3_validate_queues()	N/A	QF	Fixed
37	vmxnet3	net	qemu	6.1.50	i386	Assertion failed in vmxnet3_io_bar0_write()	N/A	QF	Fixed
38	vmxnet3	net	qemu	6.1.50	i386	Out of memory net_tx_pkt_init()	N/A	QF, VS	Fixed
39	vmxnet3	net	qemu	6.1.50	i386	Assertion failure in net_tx_pkt_reset()	N/A	QF	Fixed
40	vmxnet3	net	qemu	6.1.50	i386	eth_get_gso_type: code should not be reached	N/A	QF, VS	Fixed
41	xhci	usb	qemu	7.0.94	i386	Abort in xhci_find_stream()	56	Vi	*Fixed(us)*
42	xlnx_dp	display	qemu	7.0.91	aarch64	Abort in xlnx_dp_aux_set_command()	1	Vi	*Fixed(us)*
43	xlnx_dp	display	qemu	6.1.50	aarch64	Out of bounds read in xlnx_dp_read()	1	Vi	*Fixed(us)*
44	xlnx_dp	display	qemu	6.1.50	aarch64	Out of bounds in xlnx_dp_vblend_read()	N/A	An	Fixed
45	xlnx_dp	display	qemu	7.2.50	aarch64	Overflow in xlnx_dp_aux_push_rx_fifo()	3	Vi	Patch submitted
46	xlnx_dp	display	qemu	7.2.50	aarch64	Abort in xlnx_dp_change_graphic_fmt()	1	Vi	Patch submitted
47	xlnx_dp	display	qemu	7.2.50	aarch64	Underflow in xlnx_dp_aux_pop_tx_fifo()	1	Vi	Patch submitted
48	xlnx_dp	display	qemu	7.2.50	aarch64	Overflow in xlnx_dp_aux_push_tx_fifo()	17	Vi	Patch submitted
49	xlnx_zynqmp_can	net	qemu	7.2.50	aarch64	Fifo underflow in transfer_fifo()	2	Vi	Open
50	xlnx_zynqmp_can	net	qemu	7.2.50	aarch64	Fifo overflow in transfer_fifo()	291	Vi	Open
51	xlnx_zynqmp_qspips	spi	qemu	7.2.50	aarch64	Out of bound in xilinx_spips_write()	1	Vi	Open
52	xlnx_zynqmp_qspips	spi	qemu	7.2.50	aarch64	Underflow in xlnx_dp_aux_push_rx_fifo()	2	Vi	Open

消息内依赖注释的作用

ViDeZZo-ARP v.s. ViDeZZo-RP

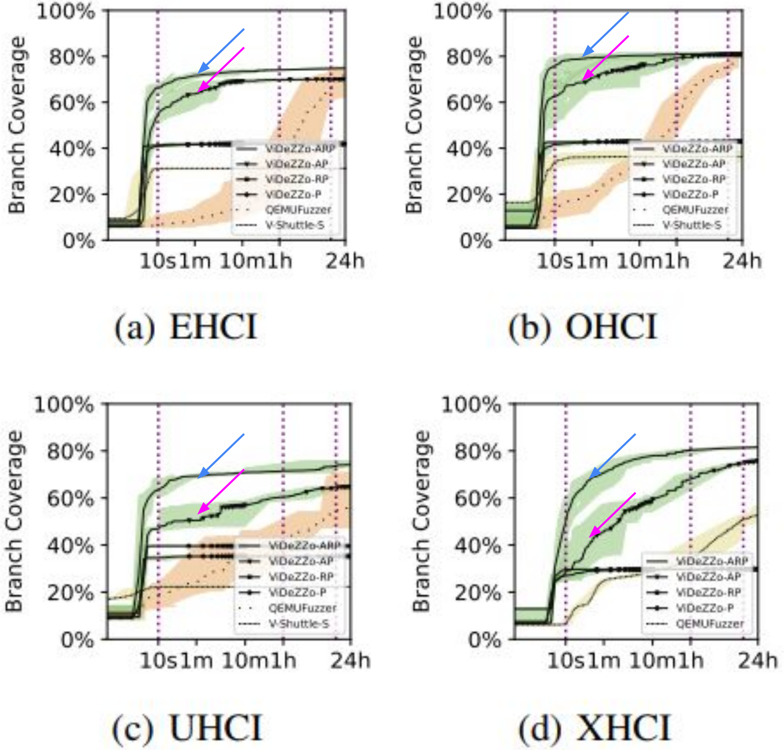
Intra-Message Dependency



多级消息突变器的作用

ViDeZZo-ARP v.s. ViDeZZo-AP

Inter-Message Dependency



(a) EHCI (b) OHCI

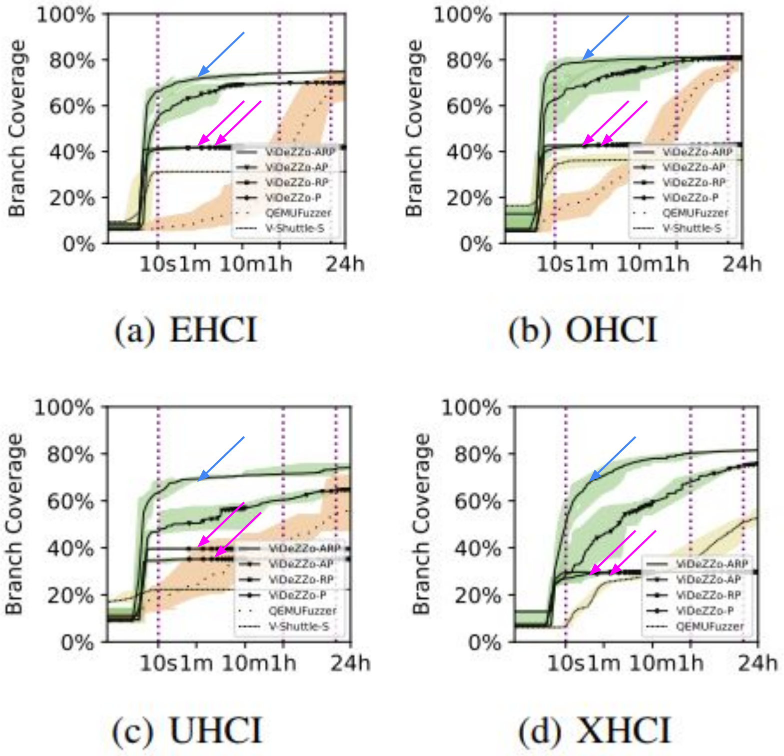
(c) UHCI (d) XHCI

多级消息突变器的作用

ViDeZZo-ARP and ViDeZZo-RP/P

Intra-Message Dependency Inter-Message Dependency

- 当支持消息内注释时，消息间变异器更加有效



(a) EHCI (b) OHCI (c) UHCI (d) XHCI

ViDeZZo: Dependency-aware Virtual Device Fuzzing

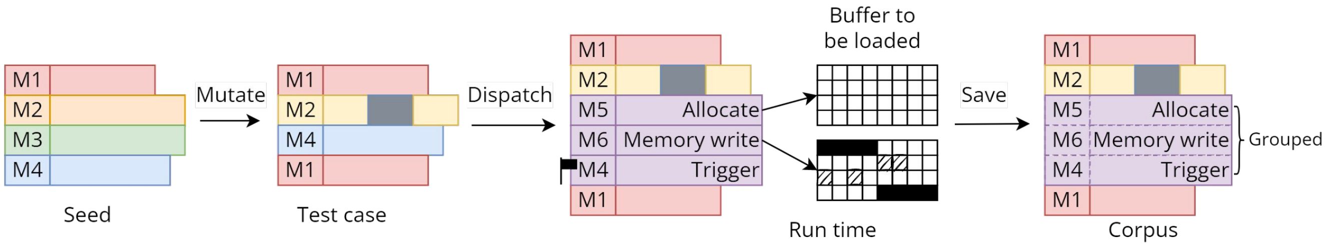
如何提高虚拟机管理程序模糊测试的效率？

虚拟设备是最大的攻击面，模糊测试时需要考虑消息内依赖和消息间依赖

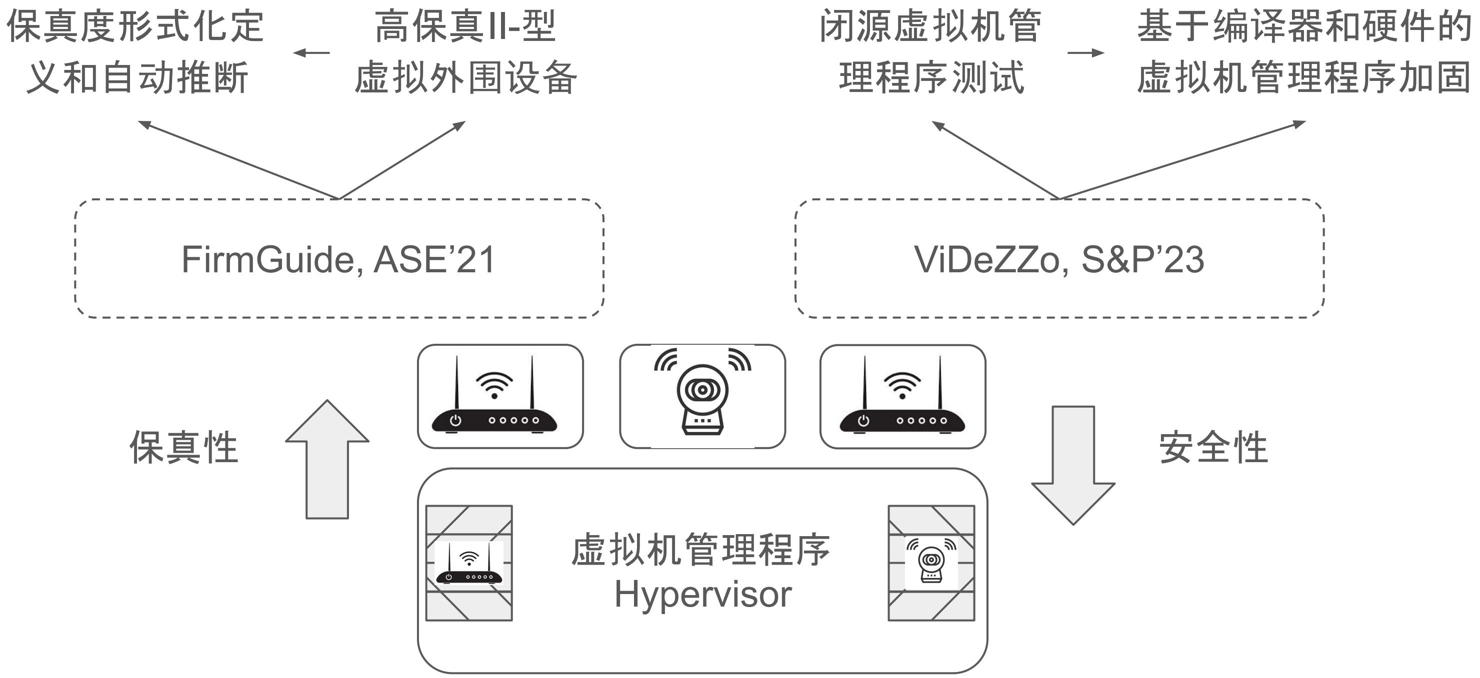
- 消息内依赖注释和多级消息突变器可以解决上述问题

ViDeZZo复现24个安全缺陷，发现了28个新的安全缺陷

自动化学习消息间依赖效率仍有待加强，将提高测试效率、简化代码实现



未来工作展望


保真度形式化定义和自动推断
高保真II-型虚拟外围设备
闭源虚拟机管理程序测试
基于编译器和硬件的虚拟机管理程序加固
FirmGuide, ASE’21
ViDeZZo, S&P’23
保真性
安全性
虚拟机管理程序
Hypervisor

总结和答疑
面向Linux物联网设备虚拟化的关键技术研究

基于模型引导内核执行的
虚拟执行环境构建研究

FirmGuide, ASE’21

基于依赖感知消息模型的
虚拟执行环境模糊测试研究

ViDeZZo, S&P’23