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

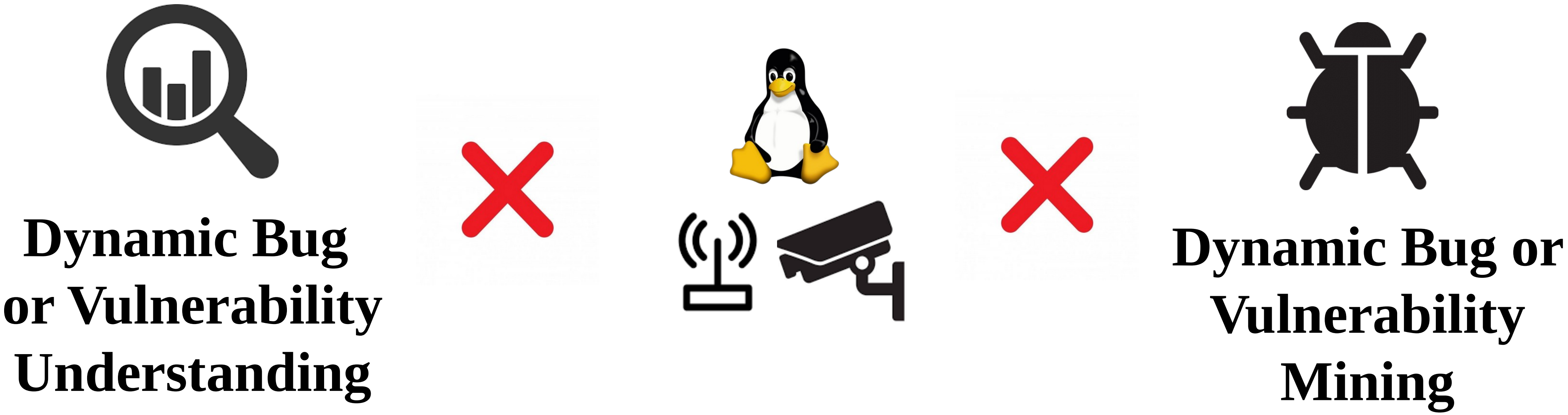
Qiang Liu[1*] Cen Zhang[2*] Lin Ma[1] Muhui Jiang[1,3] Yajin Zhou[1] Lei Wu[1] Wenbo Shen[1] Xiapu Luo[3]
Yang Liu[2] Kui Ren[1]

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

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

ASE2021

Motivation



Dynamic Bug or Vulnerability Understanding

Dynamic Bug or Vulnerability Mining

- Linux kernel with drivers inside high-end embedded firmware
- Understanding and testing abilities not easily and scalably due to hardware requirement
- **Rehosting the embedded Linux kernel with the best effort**

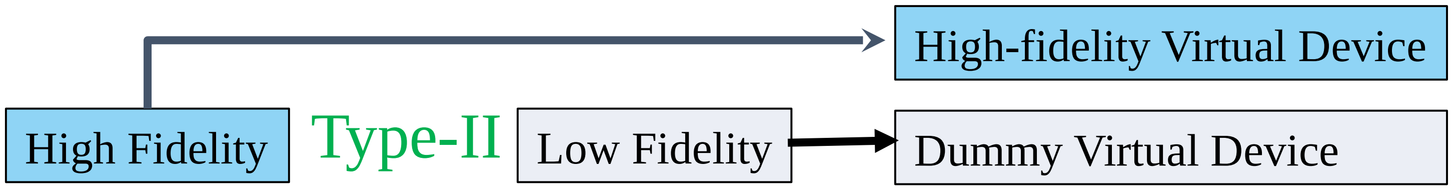
Challenge and Observation 1

SoC: plxtech,nas782x	
CPU	Arm11MPCore
Memory	up to 512M
Interrupt Controller	gic
Time-related	rps, oscillator, sysclk, plla, pllb, stdclk, twdclk
UART	ns16550a
Others	gmacclk, pcie, watchdog, sata, nand, ethernet, ehci, leds

High fidelity to make the Linux kernel functional-correct

Low fidelity for successful boot

- Numerous peripherals: Type-I High Fidelity Type-II Low Fidelity

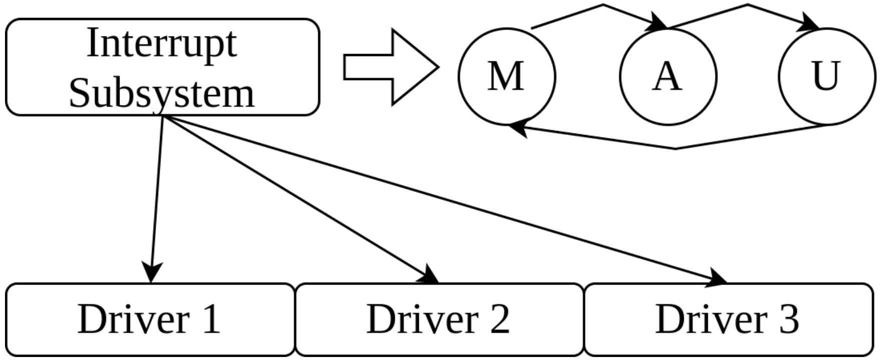


- **Classifying peripherals for a minimum best effort**

Challenge and Observation 2

Multiple models for interrupt controllers

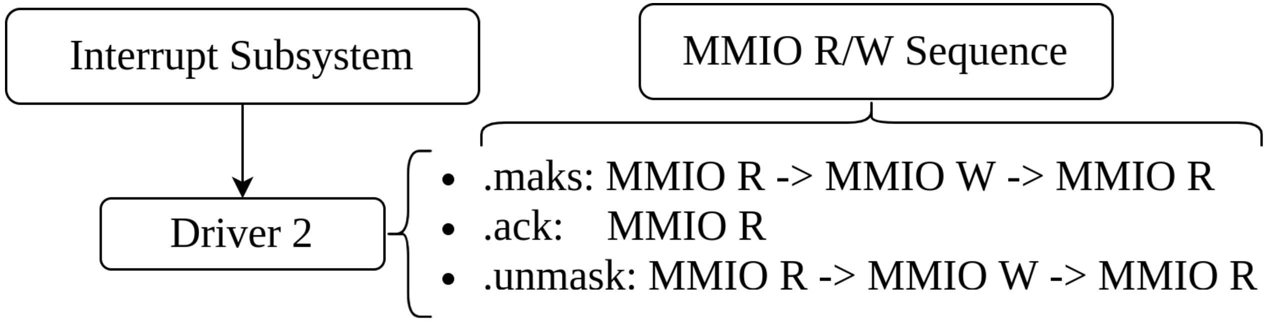
ralink-rt2880-intc
qca,ar7240-intc
marvell,orion-intc
marvell,orion-bridge-intc
arm,cortex-a9-gic
…


Interrupt Subsystem

- ***Diverse models***: Linux subsystems that hide implementation details
- **Extracting state machines from the Linux subsystems (Type-I)**

Challenge and Observation 3

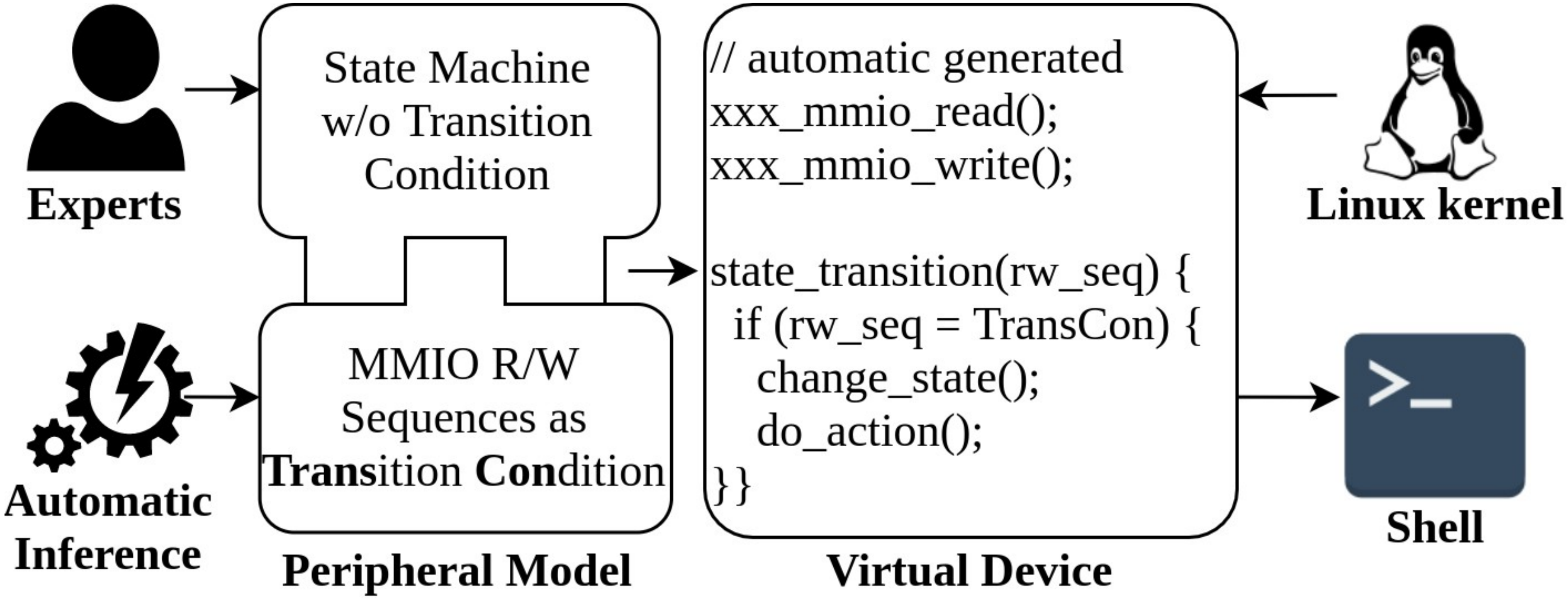
```
Mask Interrupt
    MMIO Read M -> a
    a &= flags
    MMIO Write a -> M
Load IRQ number
    MMIO Read I -> b
    switch(b)
        …
```



- .maks: MMIO R -> MMIO W -> MMIO R
- .ack: MMIO R
- .unmask: MMIO R -> MMIO W -> MMIO R

- ***Complex semantics***: Specific driver interface callbacks that embed complex semantics
- **Extracting MMIO R/W sequences**

Core Technique: Model-guided Kernel Execution

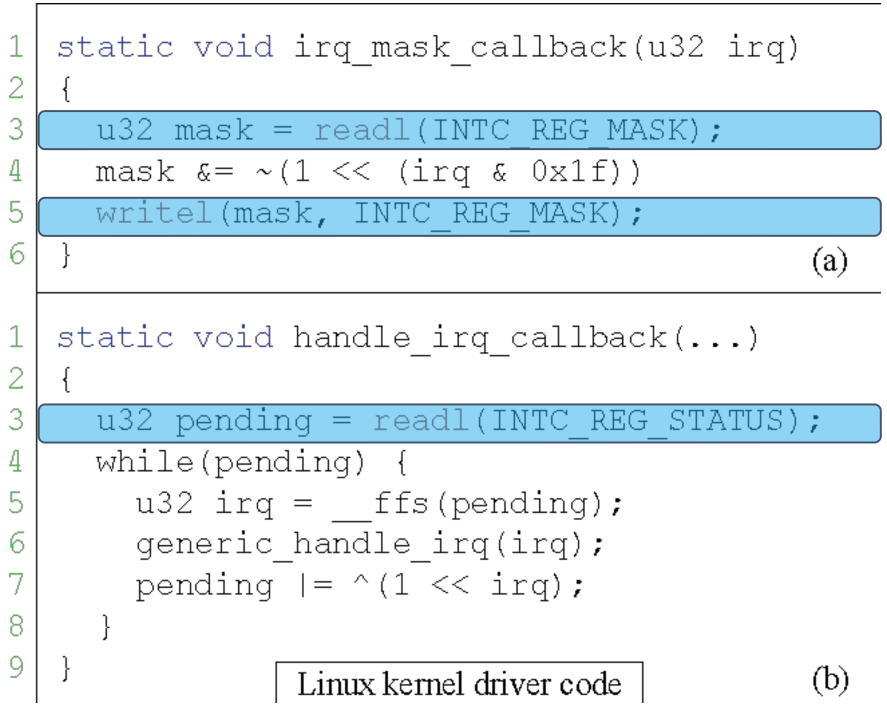

Experts
Automatic Inference
State Machine w/o Transition Condition
MMIO R/W Sequences as Transition Condition
Peripheral Model
Virtual Device
Linux kernel
Shell

```
// automatic generated
xxx_mmio_read();
xxx_mmio_write();

state_transition(rw_seq) {
  if (rw_seq = TransCon) {
    change_state();
    do_action();
}}
```

- Peripheral model = the model template (a state machine) + the model parameters (MMIO R/W sequences as transition conditions)

Model-guided Kernel Execution: Running Example



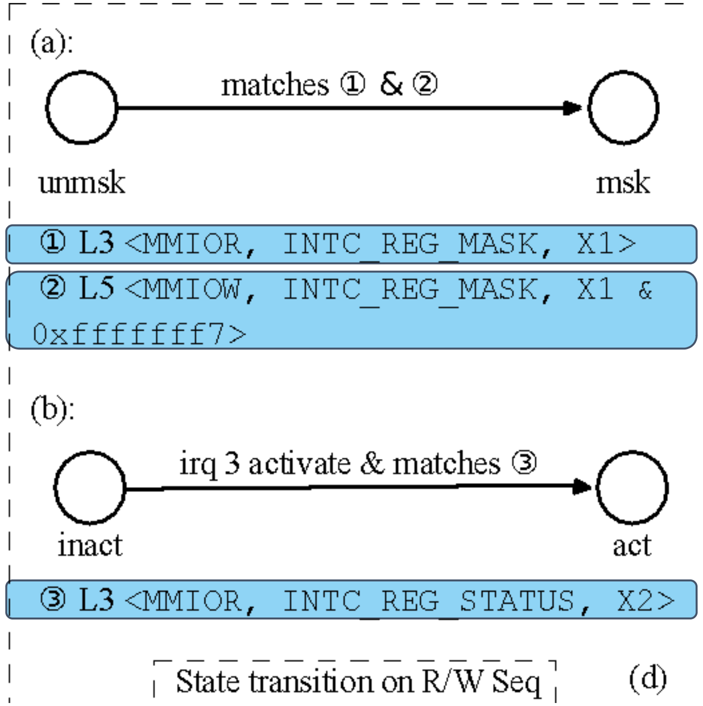
```
static void irq_mask_callback(u32 irq)
{
  u32 mask = readl(INTC_REG_MASK);
  mask &= ~(1 << (irq & 0x1f))
  writel(mask, INTC_REG_MASK);
}
```

(a)

```
static void handle_irq_callback(...)
{
  u32 pending = readl(INTC_REG_STATUS);
  while(pending) {
    u32 irq = __ffs(pending);
    generic_handle_irq(irq);
    pending |= ^(1 << irq);
  }
}
```

Linux kernel driver code

(b)



① L3 <MMIOR, INTC_REG_MASK, X1>

② L5 <MMIOW, INTC_REG_MASK, X1 & 0xffffff7>

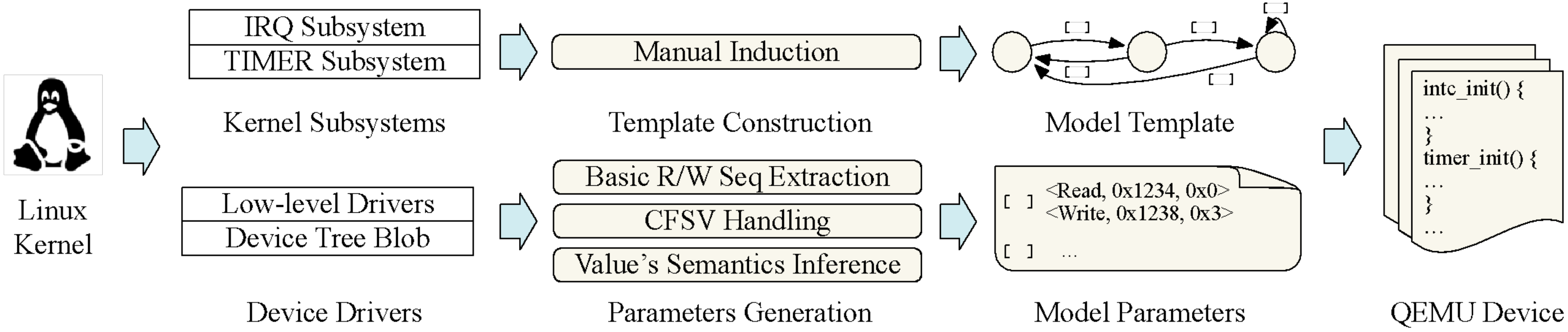
③ L3 <MMIOR, INTC_REG_STATUS, X2>

State transition on R/W Seq

(d)

- **The MMIO Read/Write sequence from Linux kernel can be recognized to drive the state machine of our emulated peripherals**

Model-guided Kernel Execution: Methodology



Linux Kernel

IRQ Subsystem

TIMER Subsystem

Kernel Subsystems

Low-level Drivers

Device Tree Blob

Device Drivers

Manual Induction

Template Construction

Basic R/W Seq Extraction

CFSV Handling

Value's Semantics Inference

Parameters Generation

Model Template

```
[ ] <Read, 0x1234, 0x0>
    <Write, 0x1238, 0x3>
[ ] ...
```

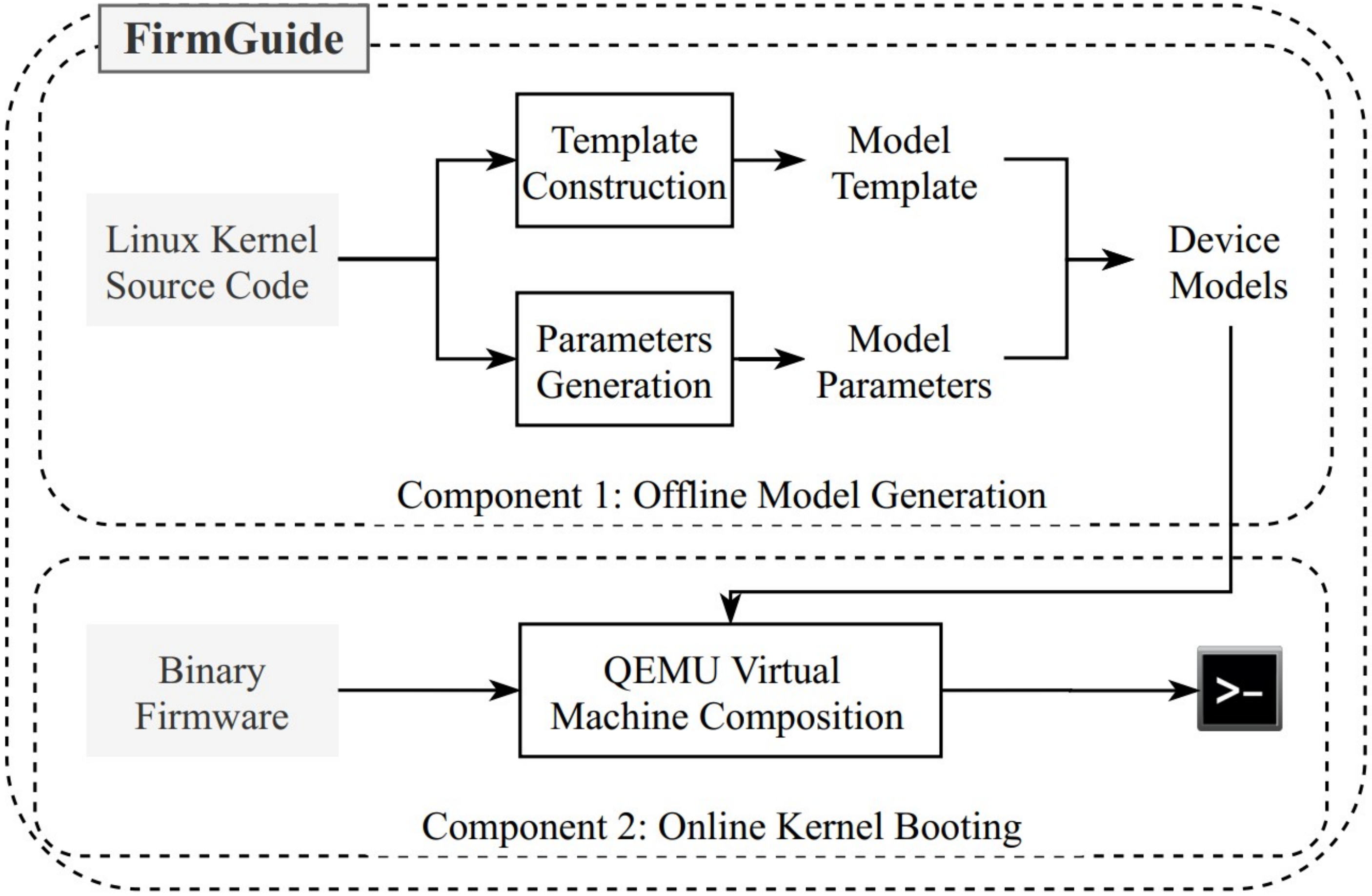
Model Parameters

```
intc_init() {
...
}
timer_init() {
...
}
...
```

QEMU Device

- **We semi-automatically build the state machine of each peripheral: a general model template (manually) plus model parameters (automatically)**

System Design and Implementation



LLVM pass for preprocess
KLEE for MMIO R/W Seq
Python for glues

Python for main logic
Template-render pattern

Evaluation

RQ 1: What peripheral models can we generate?

Type I

Family of SoCs	Interrupt Controller	Timer	First Solution (Second)	Exists CSVF (y/n)	Timer Semantics
ramips/rt305x	ralink-rt2880-intc	not necessary	1	n	-
ath79/generic	qca,ar7240-intc	not necessary	5	n	-
kirkwood/generic	marvell,orion-intc marvell,orion-bridge-intc	marvell,orion-timer	2	y	y=~x
bcm53xx/generic	arm,cortex-a9-gic	arm,cortex-a9-global-timer arm,cortex-a9-twd-timer	2,207	y	y=x1<<32+x2
oxnas/generic	arm,arm11mp-gic	plxtech,nas782x-rps-timer	914	y	y=x

Type II: # of initial values/# of Type II peripherals

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

Evaluation

RQ 2: What embedded Linux kernels can we rehost?

Subtarget	Unpack	Kernel	User Space	Shell
ramips/rt3050	4784	4784	4743 (99.14%)	4345 (90.80%)
ath79/generic	541	541	444 (82.07%)	444 (82.07%)
bcm53xx/generic	388	388	388 (100.00%)	388 (100.00%)
kirkwood/generic	330	326	324 (99.39%)	244 (74.85%)
oxnas/generic	149	149	48^ (32.21%)	48^ (32.21 %)
Overall	6192	6188	5947 (96.11%)	5469 (88.38%)

Given 6K+ firmware crossing 10 vendors, 3 architectures, and 22 Linux kernel versions, FirmGuide can successfully rehost more than 96% of them.

^The successful rate to support oxnas/generic is low because it cannot recognize our ramfs due to a unset flag.

Evaluation

RQ 3: What about the functionality of the rehosted embedded Linux kernels?

Linux Test Project: Syscall Testing

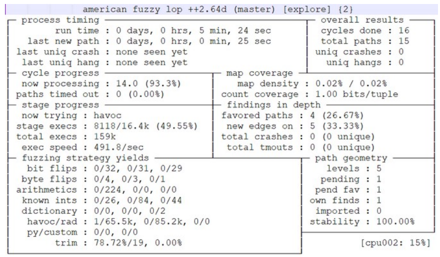
Models	**Pass**	**Skipped**	**Failed**	**Total**
Fully Generated	1049	164	46	1259
Ground Truth	1049	164	46	1259

RQ 4: What are application of FirmGuide?

CVE Reproduction and Exploit Development

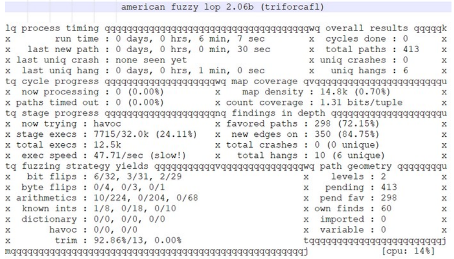
CVE ID	CVE Type	Triggering	Exploitation
CVE-2016-5195	Race Condition	N	N
CVE-2016-8655	Race Condition	Yes	Y
CVE-2016-9793	Integer Overflow	Y	N
CVE-2017-7038	Integer Overflow	Y	Y
CVD-2017-1000112	Buffer Overflow	Y	Y
CVE-2018-5333	NULL Pointer Dereference	Y	Y

Fuzzing



```
                 american fuzzy lop ++2.64d (master) [explore] {2}
┌─ process timing ─────────────────────────────────────┬─ overall results ───┐
│        run time : 0 days, 0 hrs, 5 min, 24 sec      │  cycles done : 16   │
│   last new path : 0 days, 0 hrs, 0 min, 25 sec      │  total paths : 15   │
│ last uniq crash : none seen yet                      │ uniq crashes : 0    │
│  last uniq hang : none seen yet                      │   uniq hangs : 0    │
├─ cycle progress ──────────────────┬─ map coverage ──┴─────────────────────┤
│  now processing : 14.0 (93.3%)    │    map density : 0.02% / 0.02%        │
│ paths timed out : 0 (0.00%)       │ count coverage : 1.00 bits/tuple      │
├─ stage progress ──────────────────┼─ findings in depth ───────────────────┤
│  now trying : havoc               │ favored paths : 4 (26.67%)            │
│ stage execs : 8118/16.4k (49.55%) │  new edges on : 5 (33.33%)            │
│ total execs : 159k                │ total crashes : 0 (0 unique)          │
│  exec speed : 491.8/sec           │  total tmouts : 0 (0 unique)          │
├─ fuzzing strategy yields ─────────┴───────────────┬─ path geometry ───────┤
│   bit flips : 0/32, 0/31, 0/29                    │    levels : 5         │
│  byte flips : 0/4, 0/3, 0/1                       │   pending : 1         │
│ arithmetics : 0/224, 0/0, 0/0                     │  pend fav : 1         │
│  known ints : 0/26, 0/84, 0/44                    │ own finds : 1         │
│  dictionary : 0/0, 0/0, 0/2                       │  imported : 0         │
│   havoc/rad : 1/65.5k, 0/85.2k, 0/0               │ stability : 100.00%   │
│   py/custom : 0/0, 0/0                            ├───────────────────────┘
│        trim : 78.72%/19, 0.00%                    │          [cpu002: 15%]
└───────────────────────────────────────────────────┘
```

UnicoreFuzz



```
                    american fuzzy lop 2.06b (triforcafl)

lq process timing qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqwq overall results qqqqqk
x        run time : 0 days, 0 hrs, 6 min, 7 sec       x  cycles done : 0     x
x   last new path : 0 days, 0 hrs, 0 min, 30 sec      x  total paths : 413   x
x last uniq crash : none seen yet                     x uniq crashes : 0     x
x  last uniq hang : 0 days, 0 hrs, 1 min, 0 sec       x   uniq hangs : 6     x
tq cycle progress qqqqqqqqqqqqqqqqqqqqwq map coverage qvqqqqqqqqqqqqqqqqqqqqqqu
x  now processing : 0 (0.00%)         x    map density : 14.8k (0.70%)       x
x paths timed out : 0 (0.00%)         x count coverage : 1.31 bits/tuple     x
tq stage progress qqqqqqqqqqqqqqqqqqqqnq findings in depth qqqqqqqqqqqqqqqqqqqqu
x  now trying : havoc                 x favored paths : 298 (72.15%)         x
x stage execs : 7715/32.0k (24.11%)   x  new edges on : 350 (84.75%)         x
x total execs : 12.5k                 x total crashes : 0 (0 unique)         x
x  exec speed : 47.71/sec (slow!)     x   total hangs : 10 (6 unique)        x
tq fuzzing strategy yields qqqqqqqqqqvqqqqqqqqqqqqqqqqwq path geometry qqqqqqqqu
x   bit flips : 6/32, 3/31, 2/29                     x    levels : 2          x
x  byte flips : 0/4, 0/3, 0/1                        x   pending : 413        x
x arithmetics : 10/224, 0/204, 0/68                  x  pend fav : 298        x
x  known ints : 1/8, 0/18, 0/10                      x own finds : 60         x
x  dictionary : 0/0, 0/0, 0/0                        x  imported : 0          x
x       havoc : 0/0, 0/0                             x  variable : 0          x
x        trim : 92.86%/13, 0.00%                     tqqqqqqqqqqqqqqqqqqqqqqqqj
mqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqj              [cpu: 14%]
```

TriforceAFL

Summary

Conclusion

A novel technique “Model-Guided Kernel Execution” for peripheral modeling

The first semi-automatic framework for embedded Linux kernel rehosting

Feasible dynamically understanding and mining vulnerability in embedded kernels

Discussion

Limitation and future work

Manual state machine construction for more complex peripherals

High fidelity of Type-II peripherals

Q & A

qiangliu@zju.edu.cn, cen001@e.ntu.edu.sg