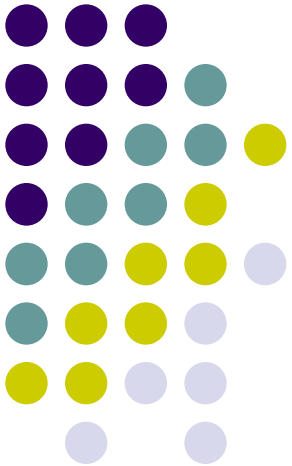
SymDrive



Department of Computer Science & Technology
Tsinghua University

- 1. Introduction
- 2. Design
- 3. Evaluation
- 4. Conclusions

清華大學

- 1. Introduction
- 2. Design
- 3. Evaluation
- 4. Conclusions

清華大學

Introduction

- Symbolic: Symbolic device, symbolic execution
- Driver: for testing driver

- =>Sym-Drive

Introduction

- 1. Static-analysis:
  - SymGen
  - A static-analysis and code transformation tool to speed testing.
- 2. Checkers:
  - Execute in the kernel.
- 3. Execution-tracing tool:
  - For logging the path of driver execution
  - Used to compare execution across different driver.

Introduction

- Purpose
  - 1. To test driver patches by thoroughly executing all branches affected by the code changes.
  - 2. A debugging tool to compare the behavior of a functioning driver against a non-functioning driver .
  - 3. A general-purpose bug-finding tool and perform broad testing of many drivers with little developer input.

Introduction

- Built on a extended S2E(Selective Symbolic Execution)
  - Why symbolic execution?
    - High coverage of code
    - Executing without device

清華大學

Introduction

- 1. Efficiency
  - Avoid path explosion
  - Fast enough to apply to every patch
- 2. Simplicity
  - Require low developer effort to test a new driver and support many device classes, buses, and OS.
- 3. Usefulness
  - Find bugs that are hard to find.

- 1. Introduction
- **2. Design**
- 3. Evaluation
- 4. Conclusions

清華大學

Design

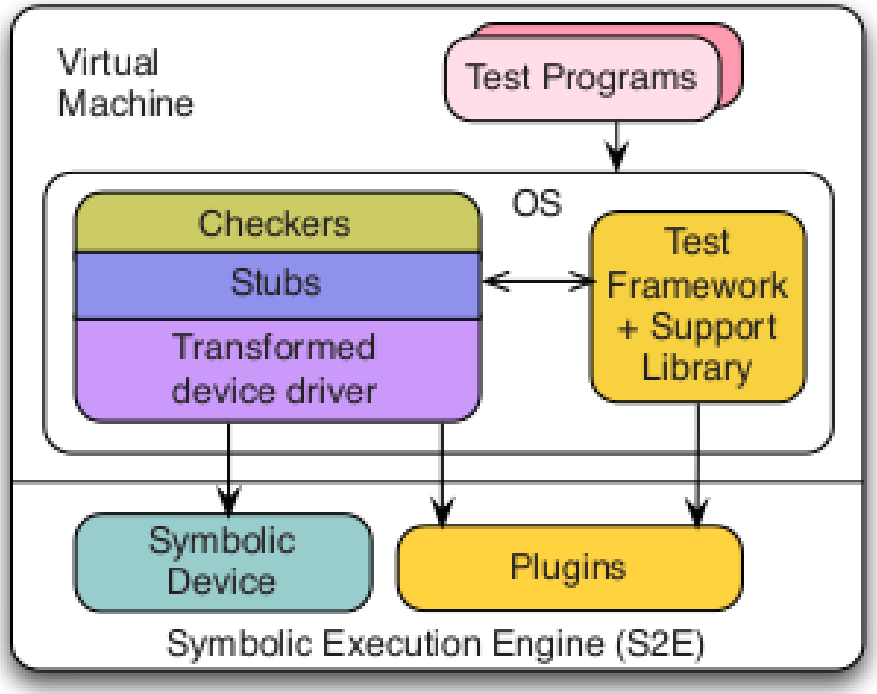


Figure 1: **The SymDrive architecture. A developer produces the transformed driver with SymGen and can write checkers and test programs to verify correctness.**

Design

Component	LoC
Changes to S^2E	1,954
SymGen	2,681
Test framework	3,002
Checkers	564
Support Library	1,579
Linux kernel changes	153
FreeBSD kernel changes	81

Table 1: **Implementation size of SymDrive.**

清華大學

Design

- 1. modified S2E
- 2. symbolic device
- 3. a test framework
- 4. SymGen
- 5. Checkers

清華大學

Modified S2E

- 1. Symbolic Devices
- 2. S2E plugin
  - Path scheduling
- 3. Opcode
  - Inserted into driver code by SymGen and invoked directly by the test framework
    - 1. control whether memory regions are symbolic when mapping data for DMA.
    - 2. influence path scheduling.
    - 3. execution-tracing

Symbolic Devices

- 1. Device Discovery
- 2. Symbolic I/O
- 3. Symbolic Interrupts
- 4. Symbolic DMA

清華大學

Test Framework

- 1. Reaching Deeply
  - Favor-success scheduling
  - Loop elision
- 2. Increasing Coverage
- 3. Execution Tracing

SymGen

- 1. Stubs
  - Pre-stub & post-stub
  - Driver/kernel switch
- 2. Instrumentation
  - Start, end, loopbody.

清華大學

SymGen

```
s2e_loop_before(__LINE__, loop_id);
while(work--) {
  tmp___17 = readb(cp->regs + 55);
  if(!(tmp___17 & 16)) goto return_label;
  stub_schedule_timeout_uninterruptible(10L);
  s2e_loop_body(__LINE__, loop_id);
}
s2e_loop_after(__LINE__, loop_id);
```

Figure 2: **SymGen instruments the start, end, and body of loops automatically. This code, from the `8139cp` driver, was modified slightly since SymGen produces preprocessed output.**

清華大學

Checkers

- Example:

```
/* Test #1 */ void __pci_register_driver_check(...) {
  if (precondition) {
    assert (state.registered == NOT_CALLED);
    set_state (&state.registered, IN_PROGRESS);
    set_driver_bus (DRIVER_PCI);
  } else /* postcondition */ {
    if (retval == 0) set_state (&state.registered, OK);
    else set_state (&state.registered, FAILED);
  }
}
```

Checkers

- 1. Detect driver/kernel interface violations
- 2. Use support library to simplify development
  - State variables
    - Track the state of the driver and current thread.
  - Tracker object
    - Record kernel objects currently in use in the driver.
  - Generic checkers

Checkers

- 1. Execution Context checker
  - E.g.
    - Verify that flags passed to memory-allocation functions such as *kmalloc* are valid in the context of the currently executing code.
- 2. Kernel API Misuse
  - State variables provide context for these tests.

Checkers

- 3.Collateral Evolutions
  - "Collateral evolutions occur when a small change to a kernel interface necessitates changes in many drivers simultaneously."
  - E.g. recent kernel network drivers:
    - *net_device->trans_start* is constant across *start_xmit* function calls
- 4. Memory Leaks
  - Use object tracker to store an allocation's address and length to ensure allocation and freeing use paired routines.

Checkers

- E.g.

```
/* Test #2 */ void __kmalloc_check
  (..., void *retval, size_t size, gfp_t flags) {
  if (precondition)
    mem_flags_test(GFP_ATOMIC, GFP_KERNEL, flags);
  else /* postcondition */
    generic_allocator(retval, size, ORIGIN_KMALLOC);
}
```

- 1. Introduction
- 2. Design
- **3. Evaluation**
- 4. Conclusions

清華大學

Driver	Class	Bugs	LoC	Ann	Load	Unld.
*akm8975**	Compass	4	629	0	0:22	0:08
*mmc31xx**	Compass	3	398	0	0:10	0:04
tle62x0*	Control	2	260	0	0:06	0:05
me4000	Data Ac.	1	5,394	2	1:17	1:04
phantom	Haptic	0	436	0	0:16	0:13
lp5523*	LED Ctl.	2	828	0	2:26	0:19
apds9802*	Light	0	256	1	0:31	0:21
8139cp	Net	0	1,610	1	1:51	0:37
8139too	Net	2	1,904	3	3:28	0:35
be2net	Net	7	3,352	2	4:49	1:39
dl2k	Net	1	1,985	5	2:52	0:35
e1000	Net	3	13,971	2	4:29	2:01
et131x	Net	2	8,122	7	6:14	0:47
forcedeth	Net	1	5,064	2	4:28	0:51
ks8851*	Net	3	1,229	0	2:05	0:13
pcnet32	Net	1	2,342	1	2:34	0:27
*smc91x**	Net	0	2,256	0	10:41	0:22
pluto2	Media	2	591	3	1:45	1:01
econet	Proto.	2	818	0	0:11	0:11
ens1371	Sound	0	2,112	5	27:07	4:48
*a1026**	Voice	1	1,116	1	0:34	0:03
ed	Net	0	5,014	0	0:49	0:13
re	Net	0	3,440	3	16:11	0:21
rl	Net	0	2,152	1	2:00	0:08
es137x	Sound	1	1,688	2	57:30	0:09
maestro	Sound	1	1,789	2	17:51	0:27

Table 2: **Drivers tested. Those in *italics* run on Android-based phones, those followed by an asterisk are for embedded systems and do not use the PCI bus. Drivers above the line are for Linux and below the line are for FreeBSD. Line counts come from CLOC [1]. Times are in minute:second format, and are an average of three runs.**

Environment:

Ubuntu 10.10

4-core Intel 2.50GHz Q9300 CPU

8GB memory

Single-threaded mode SymDrive

Operations:

1. Run SymGen over the driver.
2. Define a symbolic device and boot the SymDrive VM.
3. Load the driver with *insmod* and wait for initialization to complete successfully.
4. Execute a workload(optional).
5. Unload the driver.

清華大學

Evaluation

Bug Type	Bugs	Kernel / Checker	Cross EntPt	Paths	Ptrs
Hardware Dep.	7	6 / 1	4	6	6
API Misuse	15	7 / 8	6	5	1
Race	3	3 / 0	3	2	3
Alloc. Mismatch	3	0 / 3	3	0	3
Leak	7	0 / 7	6	1	7
Driver Interface	3	0 / 3	0	2	0
Bad pointer	1	1 / 0	0	0	1
Totals	39	17 / 22	22	16	21

Table 3: **Summary of bugs found. For each category, we present the number of bugs found by kernel crash/warning or a checker and the number that crossed driver entry points ("Cross EntPt"), occurred only on specific paths, or required tracking pointer usage.**

清華大學

Evaluation

- 39 bugs:
  - 17 bugs:fixed between 2.6.29 to 3.1.1
  - 7 bugs:unable to establish because of significant driver changes.
  - 5 bugs:submitted and confirmed.
  - Others:not in Linux kernel

Evaluation

- Test a new driver: phantom driver
  - Total time: 1h 45m
    - Configured symbolic hardware
    - Wrote a user-mode test program.
    - Executed the driver four times.
  - SymGen < 1min
  - Execute 38min

Evaluation

Driver	Touched Funcs.	Coverage	Time CPU	Time Latency
8139too	93%	83%	2h36m	1h00m
a1026	95%	80%	15m	13m
apds9802	85%	90%	14m	7m
econet	51%	61%	42m	26m
ens1371	74%	60%	*8h23m	*2h16m
lp5523	95%	83%	21m	5m
me4000	82%	68%	*26h57m	*10h25m
mmc31xx	100%	83%	14m	26m
phantom	86%	84%	38m	32m
pluto2	78%	90%	19m	6m
tle62x0	100%	85%	16m	12m
es137x	97%	70%	1h22m	58m
rl	84%	71%	13m	10m

Table 4: **Code coverage.**

8139too driver:

real hardware:77% Funcs 75% Coverage

symbolic hardware:93% Funcs 83% Coverage

清華大學

Evaluation

- Patch Testing
  - Favor-success scheduling
  - High coverage mode.

Driver	Touched Funcs.	Coverage	Time Serial	Time Parallel
8139too	100%	96%	9m	5m
ks8851	100%	100%	16m	8m
lp5523	100%	97%	12m	12m

Table 5: **Patched code coverage.**

Evaluation

- Comparison to other tools
  - S2E(8139too)
    - Coverage: 33% Func 69% Cover **vs** 93% Func 83% Cover
    - To achieve higher coverage
      - a plugin to implement a relaxed consistency model
      - 73 distinct kernel functions
    - Binary annotation
  - Static analysis tools
    - 39170 **vs** 715
  - Kernel debug support
    - A lot output
    - simplicity

- 1. Introduction
- 2. Design
- 3. Evaluation
- **4. Conclusions**

清華大學

Conclusions

- 1. Symbolic Execution
- 2. combine test framework and static analysis
- 3. lowering the barriers to testing
  - find bugs in mature driver code of a variety of types
  - allow developers to test driver patches deeply.

清華大學

Q & A?

- Thanks!